МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа экономики и управления
Кафедра «Информационные технологии в экономике»

РАБОТА ПРОВЕРЕНА
Рецензент, ведущий разработчик
ООО «Наполеон АйТи»
_____А. К. Богушов
«___»_____2018 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.т.н., с.н.с.
_____Б. М. Суховилов
«___»_____2018 г.

РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ВИЗУАЛИЗАЦИИ И АНАЛИЗА DICOM-ИЗОБРАЖЕНИЙ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ЮУрГУ – 09.04.03.2018.882 ВКР

Руководитель работы, к.т.н., доц. каф. ИТЭ _____В.А. Конов
«___»_____2018 г.

Автор работы,
Студент группы ЭУ-221
_____ _____В.Д. Колногоров
«___»_____2018 г.

Нормоконтролер,
Ст. преподаватель.
_____Е.Н. Горных
«___»_____2018 г

Челябинск 2018

АННОТАЦИЯ

Колногоров, В.Д. Разработка клиент-серверного программного обеспечения для визуализации и анализа DICOM-снимков/ В.Д. Колногоров. – Челябинск: ЮУрГУ, ЭУ-221, 2018. – 90 с., 41 ил., библиогр. список – 34 наим., 4 прил.

Разработано клиент-серверное приложение для просмотра и анализа DICOM-снимков. В клиентской части программного обеспечения реализованы выбор DICOM-каталога, отображение содержащихся в нём изображений, предобработка и формирование задачи анализа выбранного снимка. В серверной части программного обеспечения реализованы методы работы с базой данных для хранения информации о задачах анализа, запуск алгоритма анализа изображений и получение результата.

Практическая значимость разработанного программного обеспечения состоит в возможности свободного распространения, а так же использования его для обучения молодых специалистов в качестве системы поддержки принятия решений.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Проблема интеграции устройств медицинской визуализации породила в 80-90-ые годы стремление к стандартизации коммуникационных стандартов. Они охватывают широких круг задач, от интерфейса с лабораторным оборудованием до обмена информацией между отдельными клиниками. В настоящее время в мире используются различные медицинские коммуникационные стандарты: HL7, IEEE/Medix, X12, ASTM, NCPDP и другие [1].

Одновременно с этим была идентифицирована задача передачи изображений с устройств медицинской визуализации (цифровые рентгеновские системы, компьютерные, магнитно-резонансные, позитронно-эмиссионные томографы, системы ультразвуковой диагностики и т.д.). Для передачи изображений наиболее широко используется стандарт Digital Imaging and Communications in Medicine (DICOM) (рисунок 1), разработанный Американской коллегией радиологии и Национальной ассоциацией производителей электроники (ACR/NEMA). Кроме того, другие коммуникационные стандарты (HL7, X12) используют формат стандарта DICOM для передачи изображений. Стандарт широко распространён по всему миру, активно используется большинством крупных производителей медицинского оборудования, а также непрерывно развивается, имея на сегодняшний день уже третью версию. Данный формат позволяет генерировать объекты, включающие в себя множество полезной информации об объекте исследования (атрибуты пациента, демографические данные), так и о самом исследовании (информация об аппарате, на котором проводилось обследование, о мед. учреждении и персонале, о виде и условиях обследования и пр.), а также непосредственно результаты исследования – серии изображений [2].

Рисунок 1 – Пример DICOM файла,
открытого в программе Medikka Dicom Viewer

Создание DICOM-файлов не представляет трудностей, так как соответствующее прикладное программное обеспечение уже встроено в новое медицинское оборудование, в то время как возможности для интерпретации, диагностики этих файлов, и тем более – передачи их любому другому специалисту являются крайне нетривиальными. В российских реалиях возникает множество препятствий для полноценного использования DICOM-изображений [3]:

- покупка специализированного программного обеспечения, служащего для просмотра и навигации по DICOM-снимкам, на данный момент представляет собой непозволительную роскошь;
- отсутствие в наличии достаточных вычислительных мощностей, позволяющих провести изучение снимков;

Хотя, факт того, что программы такого рода положительным образом могут сказаться на качестве постановки диагноза, обоснованности принимаемых медицинских решений и ускорения лечения пациентов, несомненно, приоритетен. Вдобавок ко всему, возможность программы совершения электронного обмена информацией о проведённом исследовании пока имеет статус невозможной, несмотря на очевидные плюсы реализации таковой:

Электронный обмен позволяет обеспечить дистанционный доступ нескольких специалистов к одному и тому же медицинскому изображению, что бывает важно, например, при подготовке к консилиуму;

Электронная передача медицинских изображений из одного лечебного учреждения в другое, например, для проведения консультации, может выполняться за минуты или в крайнем случае за десятки минут, в то время как пересылка снимков обычными способами нередко занимает несколько дней;

Накопление исследований в едином источнике позволит специалистам получить комплексное видение ситуации, увеличивая вероятность постановки верного диагноза.

На основании вышесказанного **целью** выпускной квалификационной работы стала разработка клиент-серверного программного обеспечения для работы с форматом DICOM.

Исходя из цели работы, поставлены следующие **задачи**:

- обзор существующих программных продуктов для просмотра DICOM-изображений;
- выбор и анализ существующих библиотек для разработки программ, предназначенных для работы с файлами DICOM;
- реализация клиентской части программы, которая предназначается для просмотра DICOM-изображений, предобработки изображения для анализа, передачи выбранного файла на сервер для последующего анализа и загрузки полученных результатов с сервера;
- реализация серверной части программы, которая предназначается для агрегирования изображений, требующих проведения анализа, а также управлением задачами для алгоритма машинного обучения, получения обработанных изображений внутренних органов.

Объект исследования – стандартизированный формат медицинских изображений DICOM.

Предмет исследования – организация работы с медицинскими снимками формата DICOM.

Работа состоит из 3 глав, заключения, 4 приложений и списка литературы. Объем работы составляет 90 страниц. Список литературы содержит 34 наименования.

В первой главе идёт речь о сравнении отечественных и зарубежных программных разработках для работы с форматом DICOM, а также сравнение библиотек для разработки программного обеспечения, которые требуют функционала, связанного с файлами DICOM.

Во второй главе рассматривается реализация клиентской части программного обеспечения.

В третьей главе рассматривается реализация серверной части программного обеспечения в соответствии с поставленными целями и задачами.

В заключении приводятся основные результаты работы.

Практическая значимость разработанного программного обеспечения состоит в возможности использования его любыми медицинскими организациями.

1 СРАВНЕНИЕ ОТЕЧЕСТВЕННЫХ И ЗАРУБЕЖНЫХ РАЗРАБОТОК

Данная работа является логическим продолжением работы Соколова П.Е. «Разработка веб-приложения для просмотра медицинских изображений DICOM». Было предложено создать программное обеспечение, которое позволит проводить медицинским работникам скрупулёзное исследование снимков, являющее собой как тестирующую среду, так и полноценно подлежащую для исследований. Сравнение существующих разработок проводилось по критериям, представленными в таблицах приложений А и Б. Таблицы содержат в себе итоги изучения рынка программных продуктов, позиционирующих себя как средства для работы с DICOM-снимками.

Поиск существующих разработок проводился в том числе по источникам, изложенным на веб-сайте Института медицинской информатики Университета Любека (Германия), содержащего более пятидесяти вариантов программных решений для обеспечения различного функционала DICOM с подробными пояснениями [4]. Иные источники, такие как [5] и [6] позволили суммарно изучить более двухсот вариантов проектов, связанных с DICOM-визуализацией.

В результате изучения программных комплексов был сделан вывод о достаточно широком разнообразии продуктов на рынке. Однако, все программы поддаются подразделению на две группы: это отдельные программы-готовые продукты и библиотеки. Любые готовые продукты (приложение А), несмотря на широчайший функционал, не могут решить поставленных задач, так как подлежат для коммерческого использования. Для второй группы продуктов (приложение Б) характерен неполноценный набор инструментов, лишь в случае платной лицензии можно получить широкие возможности для работы с 3D-снимками.

1.1 Обзор существующих программных продуктов для работы с форматом DICOM

«RadiAnt DICOM Viewer» (рисунок 2) представляет собой мощный и высокопроизводительный инструмент, сконцентрированный на максимальном удобстве для просмотра DICOM-изображений; имеет богатую документацию. Однако, о кроссплатформенности речи быть не может: приложение работает исключительно на операционных системах семейства Windows, не имеет средств внедрения на вебе. Стоимость лицензии составляет восемьдесят евро [7].

Рисунок 2 – RadiAnt DICOM Viewer

«Inobitec DICOM Viewer» (рисунок 3) является отечественной разработкой с высоким уровнем качества. Визуализатор медицинских данных, полученных с различного оборудования, развертывается на диагностических рабочих станциях и интегрируется с PACS. Он позволяет целиком и полностью обеспечить требуемый функционал, хотя, имеет те же самые недостатки, что и у предыдущего конкурента [8].

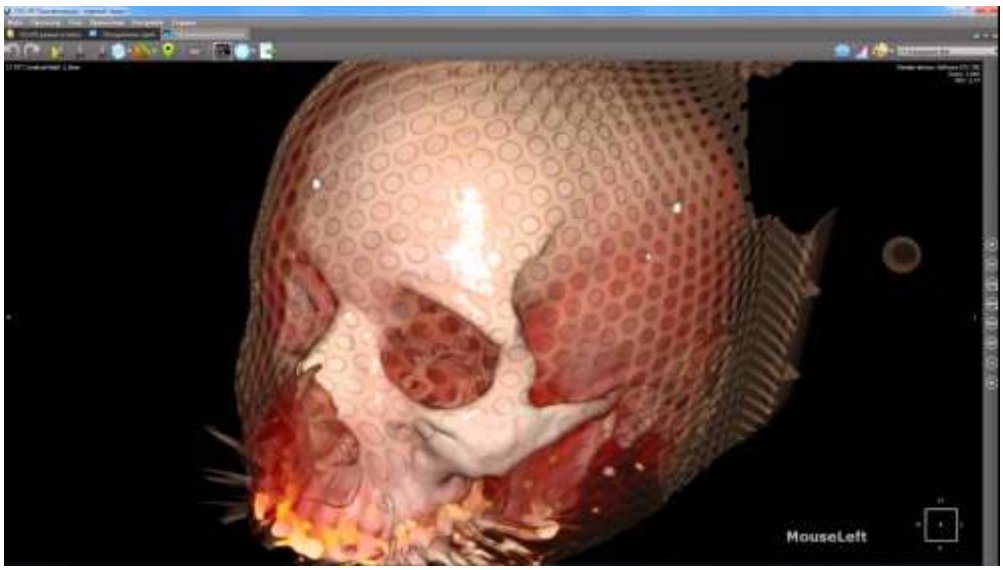

Рисунок 3 – Inobitec DICOM Viewer

«Online DICOM Viewer» (рисунок 4), как можно заключить из названия, целиком адаптированное под нужды веб-ресурса решение. Оно позволяет, загружая файлы как с компьютера, так и с облачного сервиса Google Drive, провести удовлетворительное обследование снимка лишь в 2D-плоскости. Интерфейс данной программы отвечает минимальным требованиям, быстродействие оставляет желать лучшего [9].

Рисунок 4 – Online DICOM Viewer

«Weasis DICOM Viewer» (рисунок 5) – многофункциональная веб-программа просмотра DICOM с модульной архитектурой. Создание данного

программного модуля сопряжено с потребностями нескольких клинических информационных систем в веб-доступе к радиологическим изображениям и сопутствующих мультимедийных возможностях. Задачи обработки двумерных снимков реализованы здесь достаточно полно, однако, из-за отсутствия возможностей загрузки DICOM-изображений в 3D и инструментов для их анализа данное программное обеспечение не подходит для поставленных целей [10].

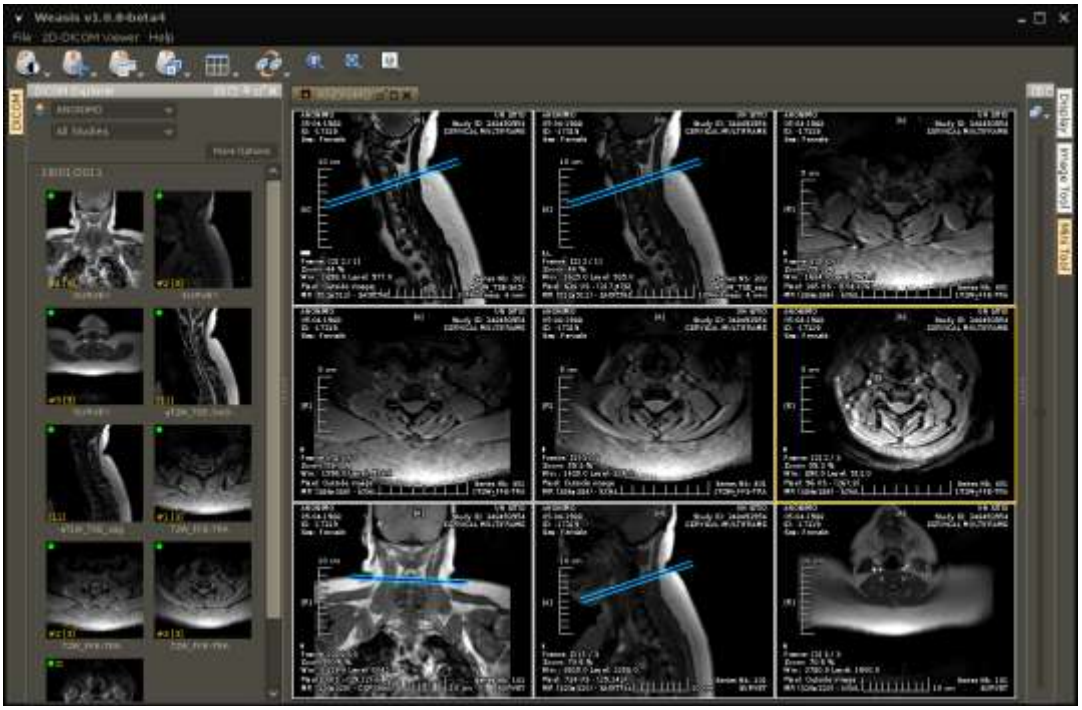

Рисунок 5 – Weasis DICOM Viewer

«Osirix» (рисунок 6) – продукт швейцарского разработчика, признанного мировым лидером в области программных решений для просмотра DICOM-изображений (имеет сертификат ISO 13485 как производитель медицинского оборудования). Функционал программы насыщен всеми необходимыми инструментами для полноценного анализа результатов медицинской визуализации. Важными недостатками этого решения необходимо отметить отсутствие кросс-платформенности, а также высокую стоимость лицензии – семьсот долларов за одну копию программы [11].

«Sante DICOM Viewer 3D» (рисунок 7), доступный в бесплатном варианте, позволяет лишь просматривать изображения, в отличие от коммерческой

версии программы. Несмотря на это, оба варианта не представляют интереса ввиду отсутствия API для внедрения в веб [12].

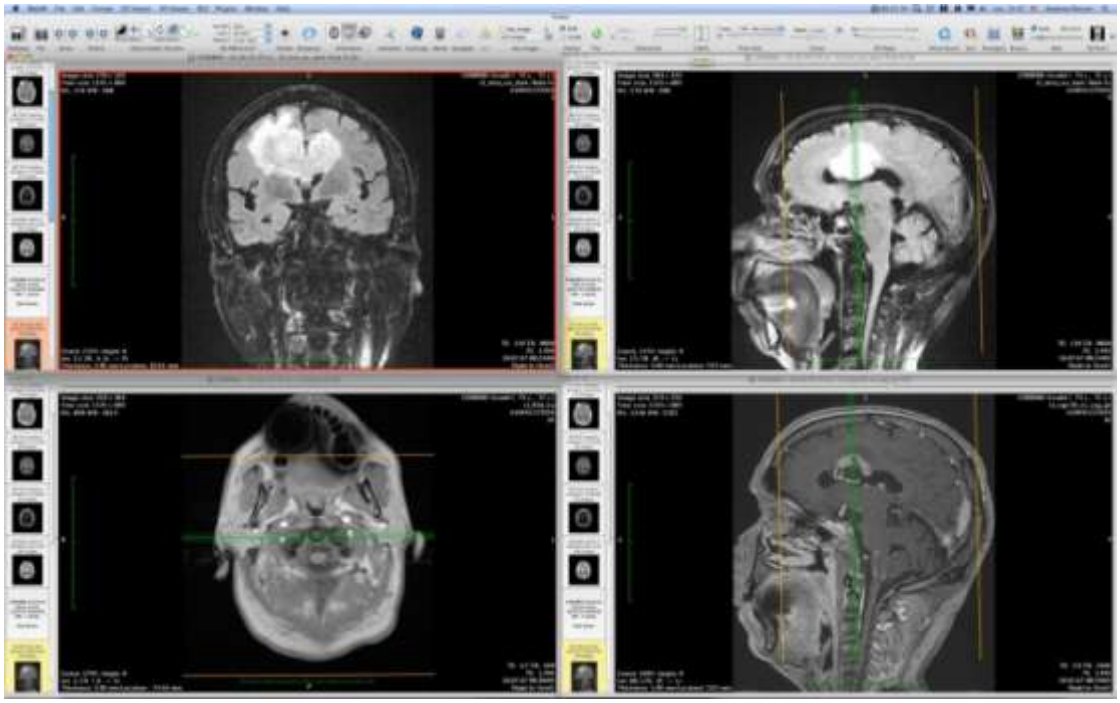

Рисунок 6 – Osirix DICOM Viewer

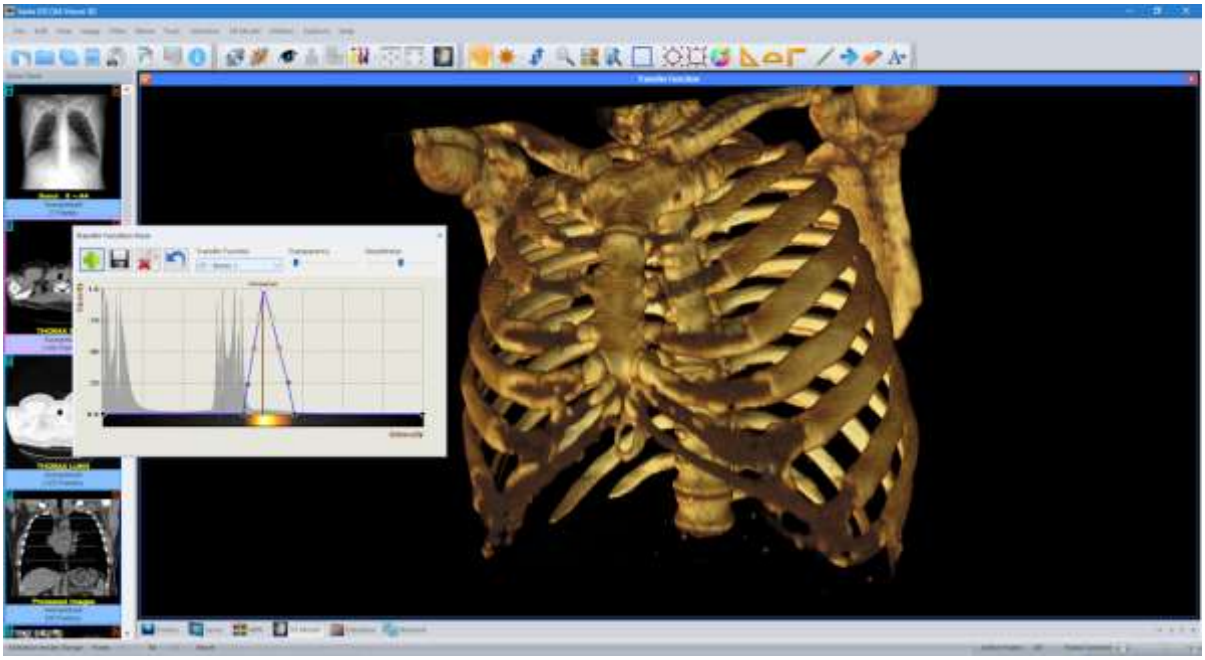

Рисунок 7 – Sante DICOM Viewer 3D

1.2 Обзор библиотек для внедрения в разрабатываемое программное обеспечение поддержку операций с форматом DICOM

Библиотеки ***«openDICOM», «Cornerstone», «FO-DICOM»*** крайне схожи по своим характеристикам: в наличии у открытых для разработки пакетов крайне скудные возможности просмотра плоских снимков. Поддержка и выпуск обновлений решений ведётся крайне медленно, и только при дополнительном финансировании проекта разработчики готовы довести программные модули до состояния, которое требуется для поставленных нами целей [13, 14, 15].

«dcm4che» (рисунок 8) – OpenSource реализация приложений, а также, библиотек и инструментов, которые разработаны с помощью языка программирования Java. Ядром «dcm4che» является именно жесткая интеграция DOCOM стандарта. «dcm4che DICOM toolkit» – инструмент создания DICOM приложений – используется разработчиками по всему миру для создания собственных медицинских решений. В настоящий момент, «dcm4che» рассматривается как один из подходящих инструментов, для решения задачи, поставленной в данной работе [10].

«DICOM Objects» **–** .Net библиотека, предоставляющая разработчикам программного обеспечения внедрять поддержку DICOM стандарта в свои продукты. ***«DICOM Objects»*** обладает всеми необходимыми расширениями для решения поставленной задачи, но эта библиотека не является бесплатной [16].

«GDCM» – библиотека с открытым исходным кодом разработанная на C++, имеющая возможность подключения к программам разрабатываемых на C++, Python, C#, Java, PHP, Perl. Данное решение не имеет нужного функционала для решения поставленной задачи [17].

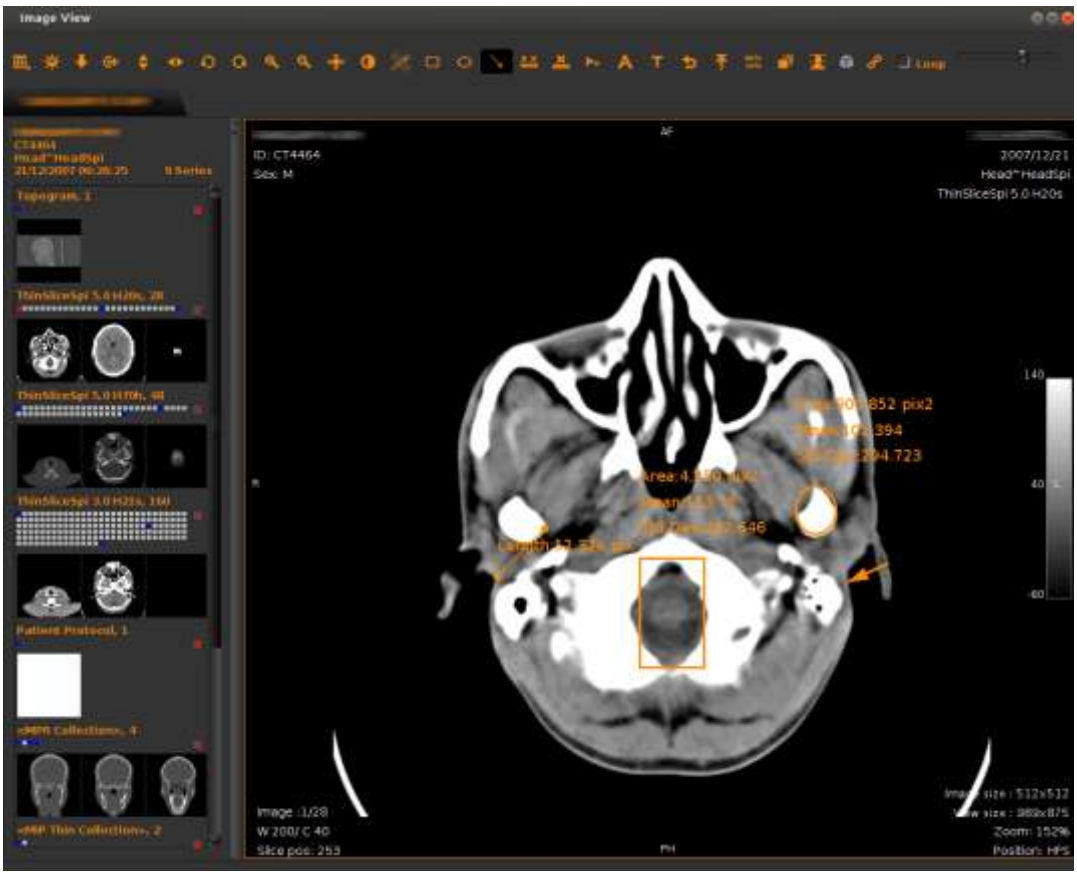

Рисунок 8 – Oviyam – DICOM Viewer, разработанный на языке программирования Java с применением библиотеки dcm4che

«LEADTOOLS» (рисунок 9) – мощное программное решение для реализации медицинского ПО с поддержкой DICOM формата. «LEADTOOLS» имеет реализацию таких методов как составление наборов данных DICOM, обмен, защита DICOM файлов, а также возможность внедрения в разрабатываемое ПО редактора DICOM файлов. «LEADTOOLS» имеет возможность интеграции в проекты на C#, VB, C/C++, WinRT, Java. Также библиотека подходит для интеграции в Web-проекты. При всех своих плюсах, «LEADTOOLS» не бесплатен, что делает его не подходящим для решения задачи, поставленной в этой работе [18].

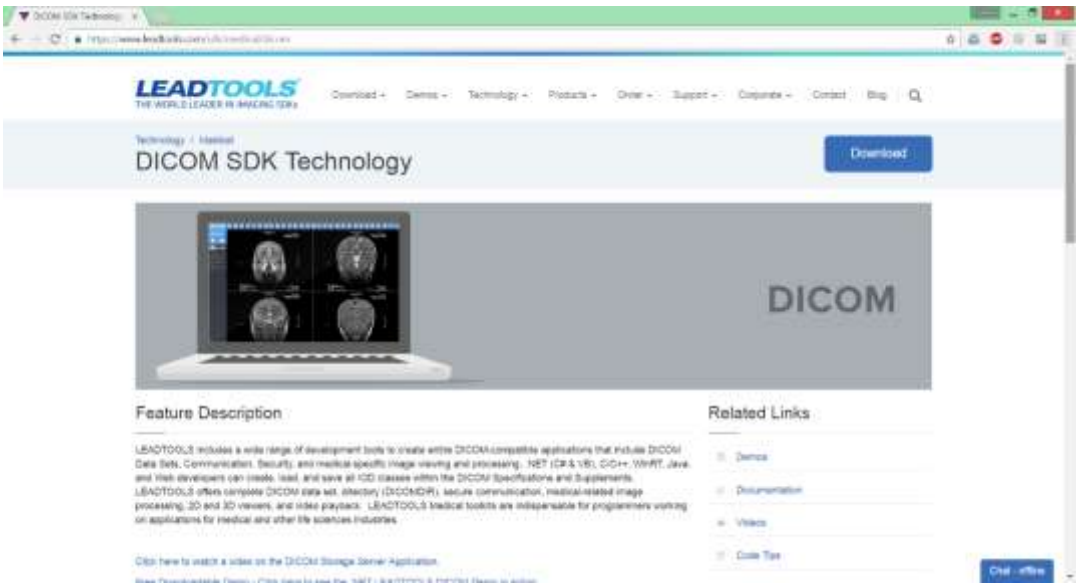

Рисунок 9 – Сайт компании LEADTOOLS

«VTK» – OpenSource библиотека, позволяющая интегрировать поддержку DICOM формата в проекты, написанные на языке C++. Библиотека предоставляет возможность визуализации и работы с DICOM файлами не только в 2D но и в 3D, что является одной из целей, поставленных в этой работе. Проект «VTK» имеет подробную документацию, примеры реализаций. В настоящий момент «VTK» рассматривается как подходящий инструмент для решения поставленной в данной работе задачи [19].

«MST DICOM SDK» – библиотека, предоставляющая полную поддержку DICOM формата, а именно чтение, редактирование, рендеринг 2D и 3D DICOM изображений. «MST DICOM SDK» не является бесплатным решением [20].

Таким образом, анализ существующих решений в области медицинской визуализации и редактирования DICOM-изображений показал, что существующее ПО хотя и обладает необходимым набором средств для организации работы с DICOM-снимками, решения агрегирующего все поставленные задачи в данной работе не было обнаружено. Наиболее функциональное ПО как правило не бесплатное, что является его главным минусом.

Анализ API для интеграции DICOM-модуля в ПО показал, что выбор библиотек (API) довольно широк, но не все проекты имеют статус Open Source: библиотеки с наиболее подходящим функционалом не бесплатны. Однако, бла-

годаря углублённому анализу были найдены библиотеки, подходящие для решения поставленной задачи.

На основании проведенного обзора существующих библиотек, для внедрения в собственный программный продукт поддержку операций с стандартизированным форматом DICOM, лучшим вариантом для решения задачи, которая в дальнейшем будет поставлена в поставленной в выпускной квалификационной работе, является библиотека VTK, а именно ее реализация для языка программирования C#.

Выводы по главе 1

Большинство программных продуктов для работы с форматов DICOM, обладающих широким функционалом, подлежат коммерческому использованию, что может повлиять на выбор медицинских учреждений. В тоже время, свободно распространяемое ПО, не всегда обладает полным функционалом для работы с форматом DICOM.

Также, не любая существующая библиотека, которая позволяет разрабатывать собственное программное обеспечение для работы с форматом DICOM, подойдет для решения поставленной задачи – многие наборы инструментария, имеющие большую базу знаний примеров и обучающего материала, распространяются исключительно на коммерческой основе, а также требуют, чтобы техническая часть, под которую планируется разрабатывать приложение, регулярно получала новые обновления.

Выбор библиотеки VTKdotNet обусловлен подходящим функционалом для решения поставленных задач, свободным исходным кодом и условиями распространения, а также поддержкой старых версий .Net Framework (от 2.0), что позволит запускать разрабатываемое программное обеспечение на большом числе персональных компьютеров.

2 РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Обзор используемых средств разработки

Так как операционные системы Windows широко распространены в медицинских организациях [21], а выбранная библиотека для разработки приложения реализована для платформы .Net, клиентская часть для работы с DICOM-снимками реализована на языке программирования «C#» в интегрированной среде разработки (IDE) Visual Studio.

Microsoft .Net Framework является так называемой программной платформой. В общих чертах можно провести аналогию с видеофайлами, которые не будут воспроизводиться если в системе не установлен нужный кодек. В данном случае видеофайл — это программа, написанная с использованием технологии .Net, а кодек — это сама платформа Microsoft .Net Framework. Причем для работы приложения, написанного на конкретной версии фреймворка, требуется установка именно этой версии, чтобы разработчик мог максимально абстрагироваться от системного окружения на компьютере пользователя, таким образом, разработчика не интересует, операционная система, разрядность и архитектура процессора компьютера пользователя и т.д. Для запуска программы достаточно чтобы под данную систему существовала и была установлена реализация .Net Framework. Для операционных систем Windows разработкой платформы занимается её создатель, компания Microsoft. Существуют также независимые реализации, прежде всего это Mono и Portable.NET, позволяющие запускать программы .Net на других операционных системах, например на Linux [22].

Windows Forms позволяет разрабатывать интеллектуальные клиенты. Интеллектуальный клиент — это приложение с полнофункциональным графическим интерфейсом, простое в развертывании и обновлении, способное работать при наличии или отсутствии подключения к Интернету и использую-

щее более безопасный доступ к ресурсам на локальном компьютере по сравнению с традиционными приложениями Windows. Windows Forms — это технология интеллектуальных клиентов для .NET Framework. Она представляет собой набор управляемых библиотек, упрощающих выполнение стандартных задач, таких как чтение из файловой системы и запись в нее. С помощью среды разработки типа Visual Studio можно создавать интеллектуальные клиентские приложения Windows Forms, которые отобра-жают информацию, запрашивают ввод от пользователей и обмениваются данными с удаленными компьютерами по сети. При выполнении пользователем какого-либо действия с формой или одним из ее элементов управления создается событие. Приложение реагирует на эти события с помощью кода и обрабатывает события при их возникновении. Windows Forms включает широкий набор элементов управления, которые можно добавлять на формы: текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы. В состав Windows Forms входят многофункциональные элементы пользовательского интерфейса, позволяющие воссоздавать возможности та-ких сложных приложений, как Microsoft Office [23].

Microsoft Visual Studio (рисунок 10) — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

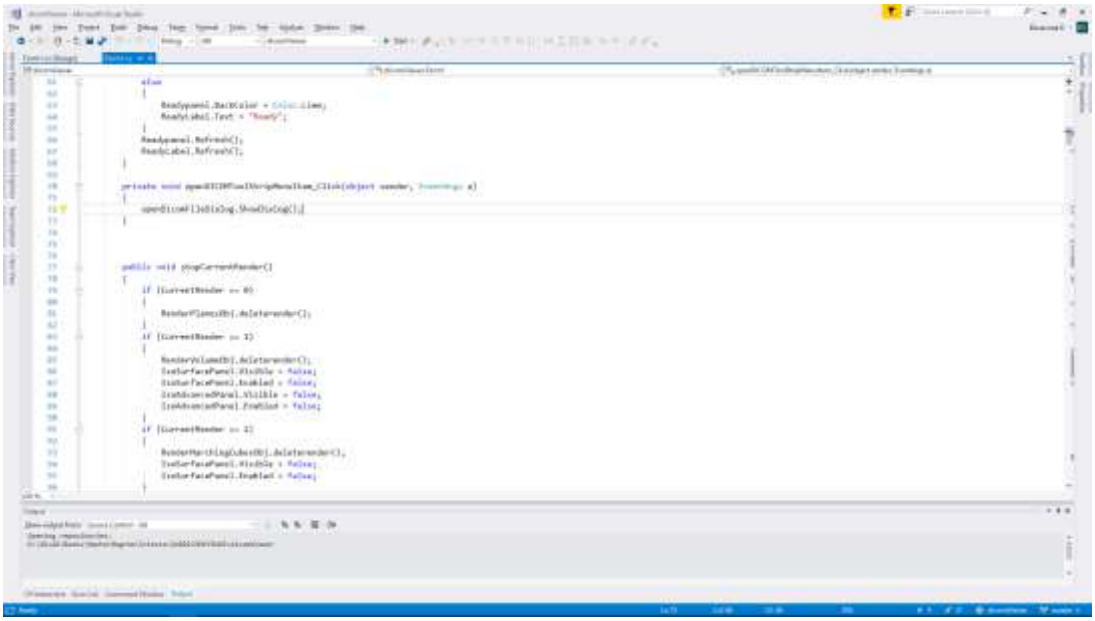

Рисунок 10 – Внешний вид рабочей области Microsoft Visual Studio 2017

Среда разработки Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server) [24].

2.2 Реализация клиентской части для просмотра DICOM-серий изображений

Клиентская часть приложения для работы с DICOM-снимками обладает следующим функционалом:

- открытие серии DICOM-снимков для просмотра срезов исследуемого органа по каждой из трех осей;
- подробный осмотр выбранного снимка и регулирование яркости и контрастности снимка для лучшего выделения обследуемого органа на нем;
- просмотр 3D-модели исследуемого органа, регулируя значения шкалы Хаунсфилда.

Для работы с DICOM-форматом в проект Visual Studio разрабатываемого приложения необходимо подключить выбранную библиотеку VTKdotNet.

Сборки бинарных файлов библиотеки помещаются в корень системного логического раздела жесткого диска в директорию с именем «VTK» (например C:\VTK). В данную директорию помещаются каталоги библиотеки:

- VTK-5.0.1-control-1.1;
- VTK-5.0.1-wrap-1.1-bin.

Также, необходимо в системной переменной «PATH» добавить пути до корневых каталогов распакованных сборок бинарных файлов (рисунок 11).

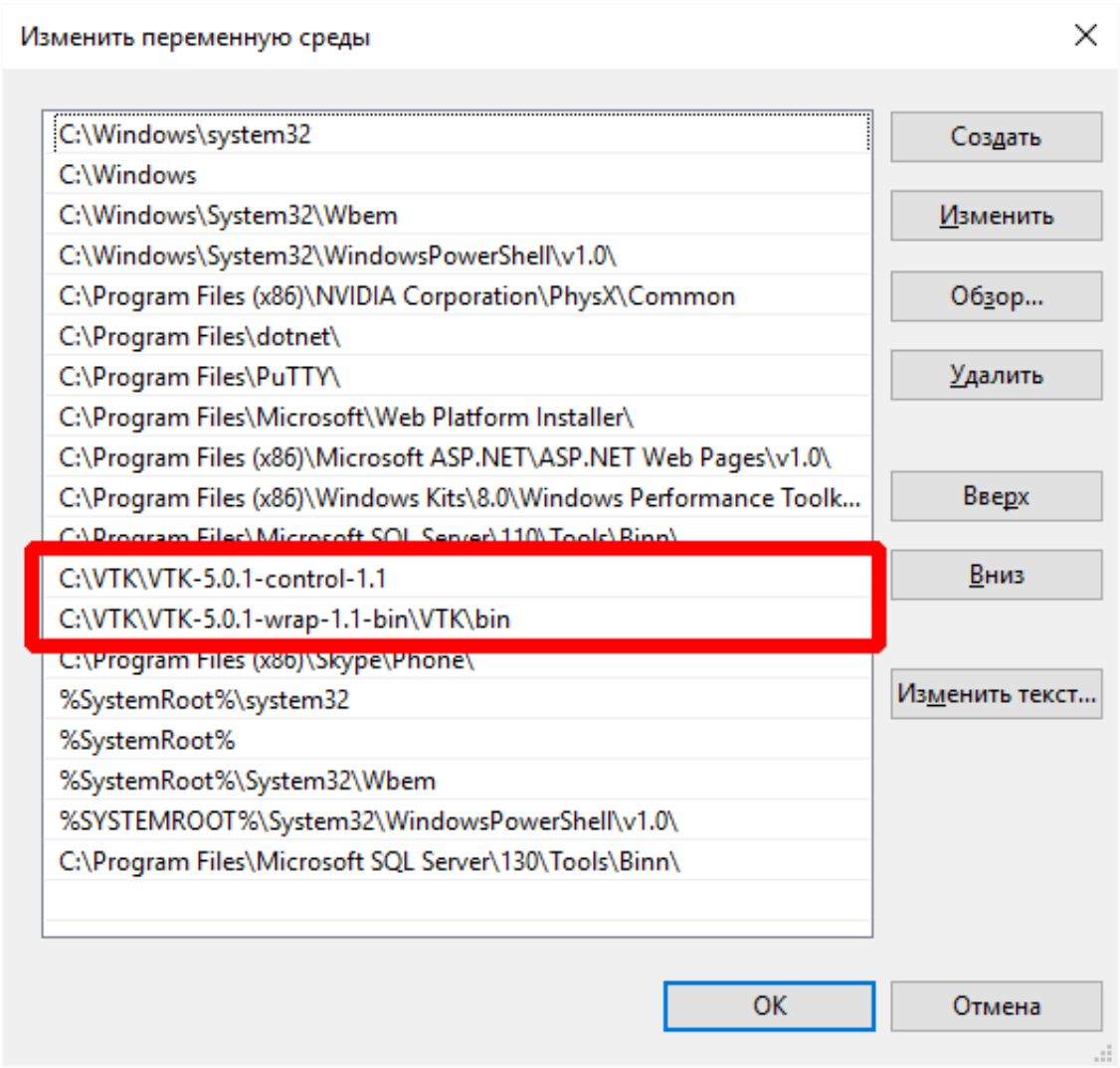

Рисунок 11 – Добавление в системную переменную PATH путей до каталогов библиотеки VTK

Далее в проекте IDE Visual Studio подключаются ссылки на необходимые бинарные сборки библиотеки (рисунок 12).

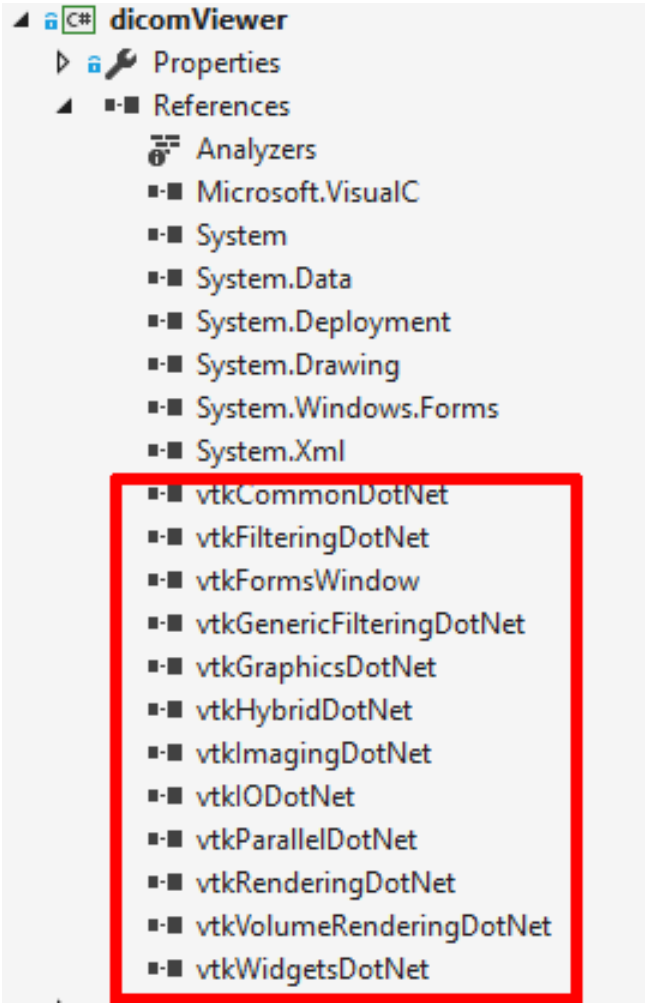

Рисунок 12 – Необходимые зависимости проекта для работы с VTK

После проделанных операций станет возможным добавить на панель инструментов Widows Forms элемент «vtkFormsWindowControl» для отображения серий DICOM-снимков (рисунок 13).

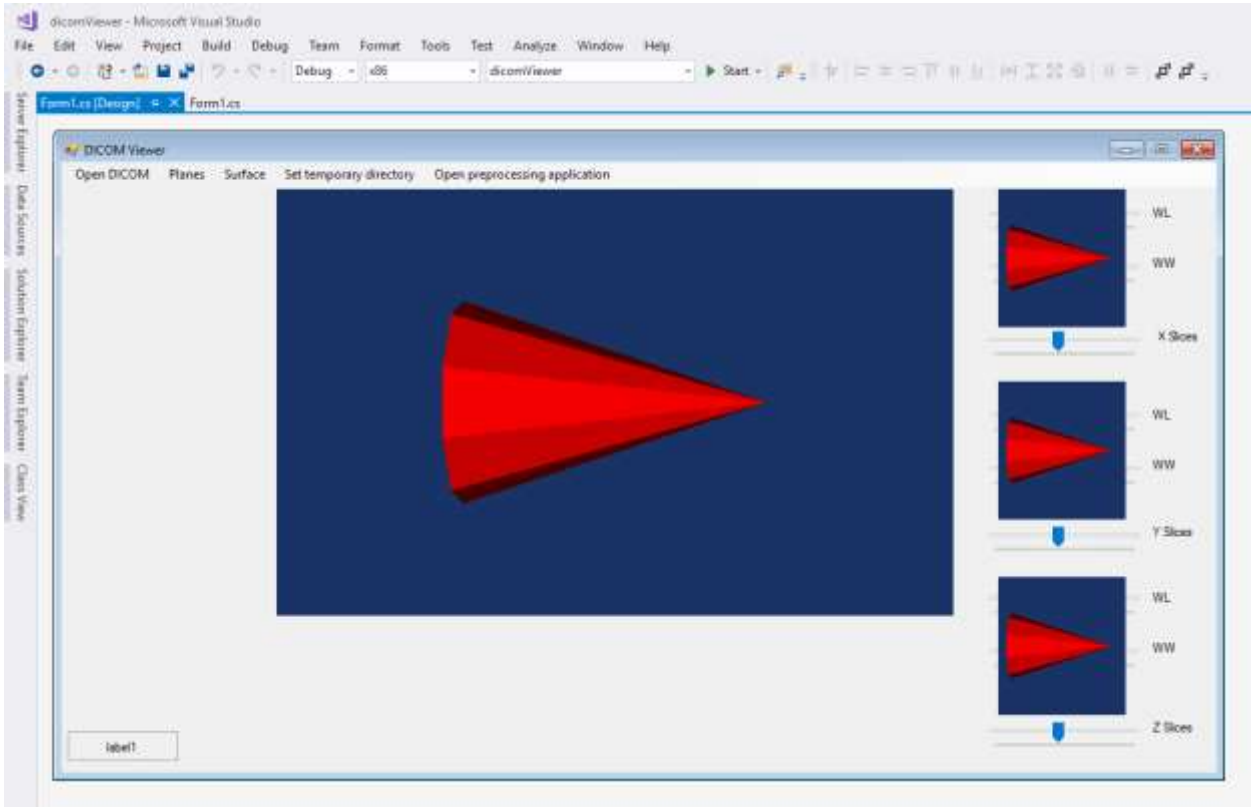

Рисунок 13 – Помещенные на форму элементы «vtkFormsWindowControl» в конструкторе Visual Studio

Работа с приложением начинается с выбора директории, в которой находятся результаты медицинского обследования, хранящиеся в формате «.dcm» (сокращение от DICOM). Выбор каталога начинается с нажатия на кнопку «Open DICOM». После выбора первого снимка в директории, начнёт работу алгоритм считывания серии DICOM-снимков. (листинг 1)

Листинг 1 – Подтверждение выбора DICOM-каталога

```
1   stopCurrentRender();
2   Program.IniObj.IniWriteValue("Info",
3               "dicom_folder", Path.GetDirectoryName(
4               openDicomFileDialog.FileName ));
5   string tempfile =
6               openDicomFileDialog.FileName;
7   tempfile = tempfile.Replace("/", "\\");
```

Окончание листинга 1

```
8   string[] fileparts = tempfile.Split('\\');
9   string path = "";
10  for (int i = 0; i < fileparts.Length - 1; i++)
11  {
12      path += fileparts[i] + "\\";
13  }
14  SelectedPath = path;
15  readtype = 1;
16  openDICOMToolStripMenuItem.Enabled =
17      true;
18  LoadTimer.Enabled = true;
19  ShowBusy(true);
```

В строке 1 срабатывает метод «stopCurrentRender», освобождающий ресурсы со всех элементов «vtkFormsWindowControl», чтобы на них было возможным отобразить новую выбранную серию DICOM-снимков.

Со строки 3 по 4 в ini-файл приложения записывается переменная, содержащая путь до выбранного каталога. Также в переменную «SelectedPath» записывается путь до выбранного каталога (строка 14), которая используется для передачи расположения файлов методу, выполняющего загрузку DICOM-серий.

В строке 18 запускается таймер, в событии «Tick», которого происходит загрузка серий снимков (листинг 2).

В строке 19 вызывается метод, устанавливающий флаг «Занят», который используется для программной и графической (на пользовательском интерфейсе) идентификации, что приложение в данный момент производит загрузку данных.

Листинг 2 – Процесс обновления элементов формы

```
1   ShowBusy(true);
2   VolumeFileObj.readVTKDicom(SelectedPath);
3   UpdateSliceNumber();
4   SiteplanesObj.SetWindows(vtkFormsWindowControlX.
5                 GetRenderWindow(),
6                 vtkFormsWindowControlY.GetRenderWindow(),
7                 vtkFormsWindowControlZ.GetRenderWindow());
8   SiteplanesObj.SetParameters(slizex, slizey, slizez);
9   SiteplanesObj.rendersiteplanes(VolumeFileObj.VoxelData);
10
11  trackBarBrightnessX.Minimum =
12     (int)(SiteplanesObj.rangeXMin*2.5);
13  trackBarBrightnessX.Maximum =
14     (-1) * trackBarBrightnessX.Minimum;
15  trackBarBrightnessX.Value = 0;
16
17  trackBartrackBarContrastX.Maximum =
18     (int)SiteplanesObj.rangeXMax*15;
19  trackBartrackBarContrastX.Value =
20     (int)SiteplanesObj.windowX;
21
22  trackBarBrightnessY.Minimum =
23     (int)(SiteplanesObj.rangeYMin * 2.5);
24  trackBarBrightnessY.Maximum =
25     (-1) * trackBarBrightnessY.Minimum;
26  trackBarBrightnessY.Value = 0;
27  trackBarContrastY.Maximum =
28     (int)SiteplanesObj.rangeYMax * 15;
29  trackBarContrastY.Value =
30     (int)SiteplanesObj.windowY;
31  trackBarBrightnessZ.Minimum =
32     (int)(SiteplanesObj.rangeZMin * 2.5);
33  trackBarBrightnessZ.Maximum =
34     (-1) * trackBarBrightnessZ.Minimum;
35  trackBarBrightnessZ.Value = 0;
36  trackBarContrastZ.Maximum =
37     (int)SiteplanesObj.rangeZMax * 15;
38  trackBarContrastZ.Value = (int)SiteplanesObj.windowZ;
39
40  RenderPlanesProcess();
41  ShowBusy(false);
42  GC.Collect();
```

В строке 1 флаг занятости приложения устанавливается в значение true, для графического отображения.

В строке 2 вызывается метод, использующий средства библиотеки VTKdotNet для загрузки dcm-файлов из ранее выбранной директории (листинг 3).

После загрузки данных, необходимо обновить переменные, в которых хранится число слайсов по каждой из трех измерений обследуемой части тела (строка 3).

В строках с 4 по 9 в объект, отвечающий за отображение DICOM-снимков раздельно по каждой из осей X, Y и Z в элементах «vtkFormsWindowControl», помещаются ссылки на каждое окно отображения, количество слайсов по всем измерениям, а также данные, полученные от чтения DICOM-снимков средствами VTKdotNet, необходимые для построения 3D-модели положения слайсов исследуемой части тела (рисунок 14).

В строках с 11 по 20 устанавливаются минимальные и максимальные значения для элементов trackBar библиотеки Windows Forms, которые регулируют яркость и контрастность выбранного DICOM-слайса оси X.

Аналогично в строках 22 по 39 определяются максимальные и минимальные значения для trackBar осей Y и Z.

Далее, в строке 41 запускается рендер положения слайсов в главном окне «vtkFormsWindowControl» средствами VTKdotNet.

В строке 42 флаг занятости приложения устанавливается в значение false для графического отображения.

В строке 43 сборщик мусора .Net фреймворка автоматически освобождает неуправляемые участки памяти.

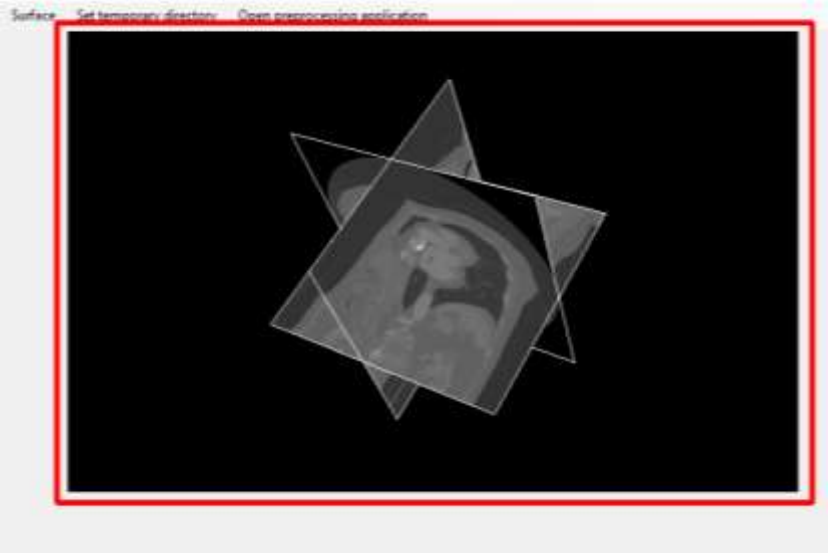

Рисунок 14 – Трехмерная визуализация отображения измерений DICOM-снимков

Листинг 3 – Загрузка данных из DICOM-файлов

```
1   vtk.vtkDICOMImageReader DICOMreader =
2      new vtk.vtkDICOMImageReader();
3   DICOMreader.SetDirectoryName(dirname);
4   DICOMreader.SetDataOrigin(0, 0, 0);
5   DICOMreader.Update();
6
7   vtk.vtkImageChangeInformation imageChangeInformation =
8      new vtk.vtkImageChangeInformation();
9   imageChangeInformation.SetInput(DICOMreader.GetOutput());
10  imageChangeInformation.CenterImageOn();
11  imageChangeInformation.Update();
12  VoxelData = imageChangeInformation.GetOutput();
13
14  spacing = VoxelData.GetSpacing();
15  int numberofpixels = VoxelData.GetNumberOfPoints();
16
17  dimensions = VoxelData.GetDimensions();
18  metaScalarRange = VoxelData.GetScalarRange();
19  double[] origin = VoxelData.GetOrigin();
20  GC.Collect();
```

С помощью объекта библиотеки VTKdotNet – vtkDICOMImageReader, в строках с 1 по 5 устанавливаются корневой каталог выбранной серии DICOM-изображений и начало отсчёта системы координат снимков.

В строках с 7 по 12 выгружаются с необходимые данные для дальнейшего отображения 3D-модели исследуемой части тела.

В строках с 14 по 19 устанавливаются переменные для управления слайсами каждого из трех измерений DICOM-изображений.

В строке 20 сборщик мусора .Net фреймворка автоматически освобождает неуправляемые участки памяти.

После загрузки данных на пользовательском интерфейсе станет возможным взаимодействие с сериями слайсов и 3D-моделью DICOM-изображений.

На главном элементе «vtkFormsWindowControl» по-умолчанию отображается объемное положение слайсов DICOM-серии.

Справа, в элементах «vtkFormsWindowControlX», «vtkFormsWindowControlY» и «vtkFormsWindowControlZ» доступен просмотр 2D-слайсов каждого из трёх измерений исследуемой части тела соответственно (рисунок 15).

Рисунок 15 – Двумерное отображение каждого из измерений DICOM-снимков

Переключение между типами рендера главного элемента «vtkFormsWindowControl» (рендер слайсов и рендер объемной модели) осуществляется элементами управления ToolStripMenuItem – Planes и Surface (для рендера слайсов и объёма соответственно) находящимися сверху основной формы (листинг 4).

Листинг 4 – Запуск отображения 3D-модели

```
1   private void surfaceToolStripMenuItem_Click(object send-
    er, EventArgs e)
2   {
3               if (CurrentRender != 1)
4               {
5                   stopCurrentRender();
6                   CurrentRender = 1;
7                   RenderIsoProcess();
8               }
9   }
10
11  private void RenderIsoProcess()
12  {
13              ShowBusy(true);
14
15              RenderVolumeObj.setisovalue(Volume
16              FileObj.metaScalarRange);
17
18              RenderVolumeObj.SetWindows(
19              vtkFormsWindowControlMain.GetRenderWindow());
20              RenderVolumeObj.VolumeRenderISO(
21              VolumeFileObj.VoxelData);
22
23              GC.Collect();
24              ShowBusy(false);
25  }
```

Со строки 1 по строку 9 представлена функция, срабатывающая на нажатие элемента управления surfaceToolStripMenuItem, для начала рендера объемной фигуры.

В строке 3 проверяется флаг CurrentRender, хранящий текущий тип рендера объемного отображения исследуемой части тела. Если этот флаг установлен в значение отличное от 1, то это определяет, что текущий рендер – отображение слайсов, и требуется произвести действия для отображения объемной модели:

- останавливается отображение рендера сплайнов (строка 5);
- флаг, хранящий тип текущего отображения модели устанавливается в значение 1 – рендер объемной фигуры (строка 6);

- начинает работу метод, начинающий рендер объемной фигуры (строка 7).

Со строки 11 по строку 25 представлен метод рендера объемной фигуры.

Метка занятости основного потока переходит в состояние true для графического отображения (строка 13), в объекте рендера устанавливается значение плотности модели (строка с 15 по 16), в строке 14 сообщается выходное окно для модели, далее, вызывается метод рендера объемной фигуры с сообщением данных воксельного рендера (строка с 18 по 21), и метка занятости основного процесса возвращается в состояние false (строка 24).

В результате получаем в окне вывода отображение объемной фигуры (рисунок 16).

Рисунок 16 – Отображение объемной фигуры

DICOM-снимки несут в себе также информацию о плотности модели. Для переключения этого параметра предусмотрены элементы управления trackBar и textBox для выбора значения плотности модели. Эти значения плотностей совпадают с плотностями шкалы Хаунсфилда (количественная шкала рентгеновской плотности (CT Values) (нужна какая-то сслыка)) (таблица 1),

примеры отображения модели с разными плотностями представлены на рисунках 17 и 18.

Таблица 1 – Значения плотностей шкалы Хаунсфилда

Вещество	**HU**
Воздух	−1000
Жир	−120
Вода	0
Мягкие ткани	+40
Кости	+400 и выше

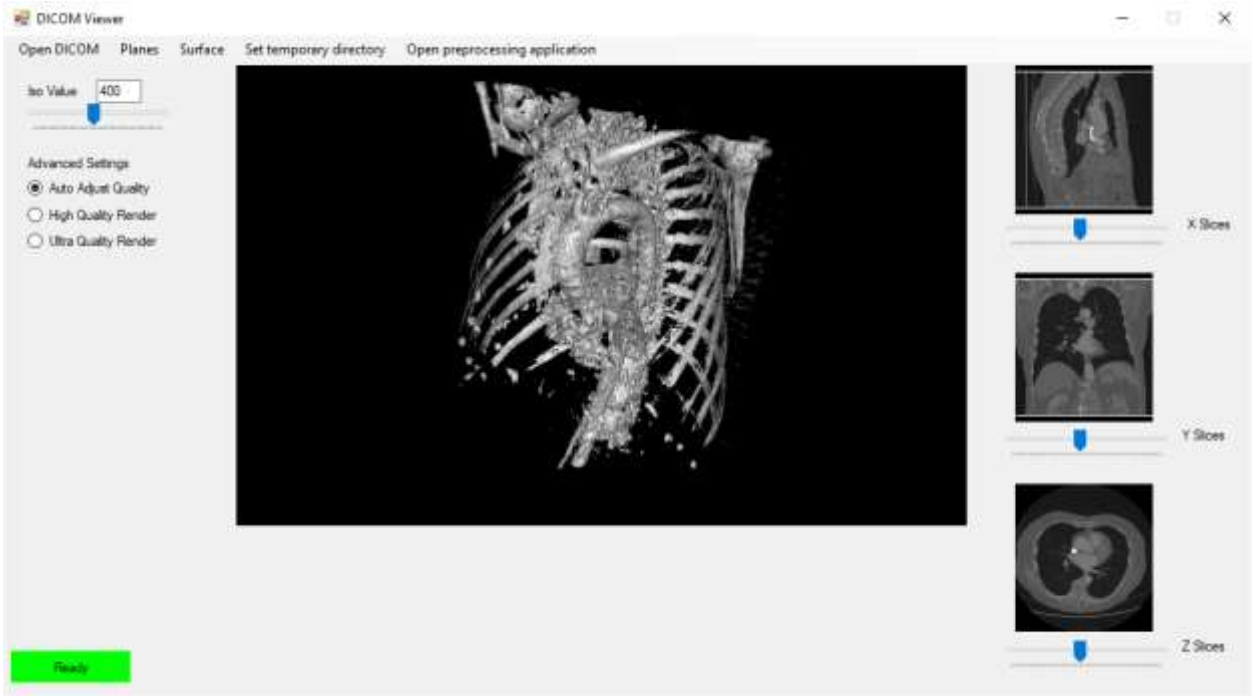

Рисунок 17 – Отображение модели со значением плотности 400, что соответствует костям

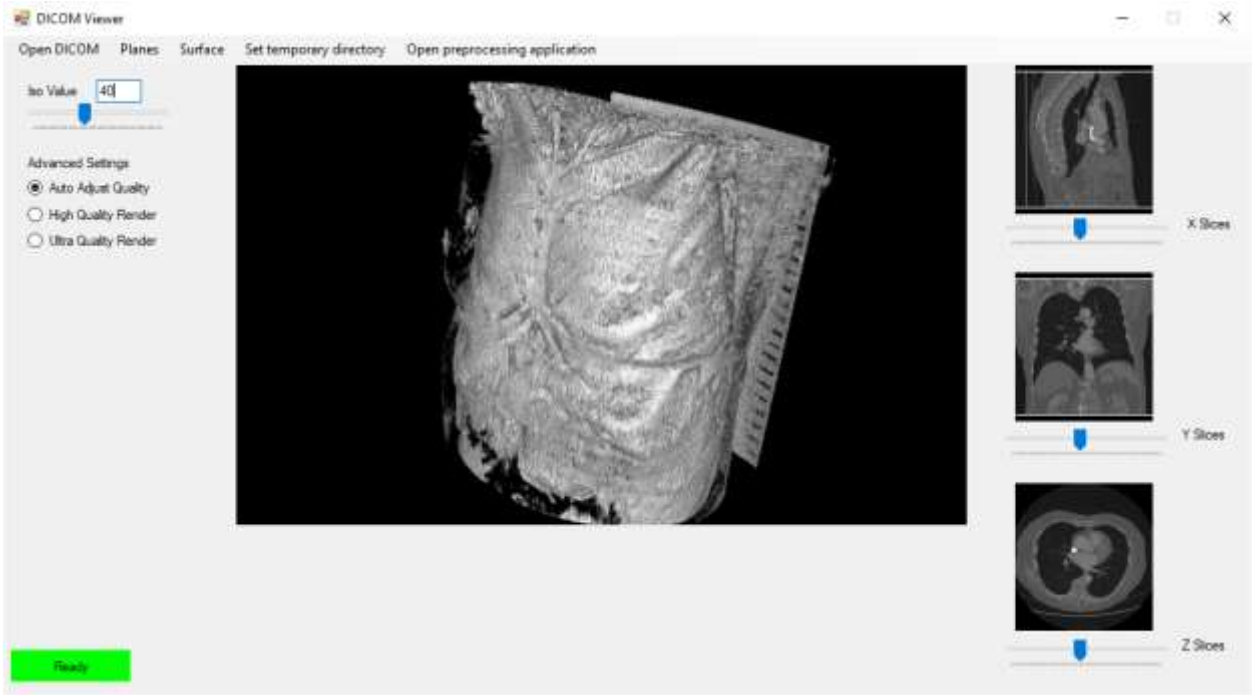

Рисунок 18 – Отображение модели со значением плотности 40,
что соответствует мягким тканям

Для подробного просмотра каждого из слайсов DICOM-серии предусмотрен переход к большему окну 2D-отображения. Для активации просмотра требуется дважды щелкнуть левой кнопкой мыши по одному из трех элементов управления «vtkFormsWindowControlX», «vtkFormsWindowControlY» или «vtkFormsWindowControlZ». В результате, посередине рабочего окна откроется 2D-представление выбранного измерения. Справа, на месте первоначальной миниатюры, станут доступны два элемента trackBar, регулирующие яркость и контрастность выбранного слайса (рисунок 19). Это требуется для дельнейшего взаимодействия с алгоритмом анализа DICOM-изображений.

Рисунок 19 – Подробный просмотр выбранного слайса

2.1 Реализация клиентской части для предобработки выбранного DICOM-снимка

Следующий шаг рабочего процесса ПО, разрабатываемого в рамках данной работы, является предобработка выбранного DICOM-слайса. Решение об реализации алгоритма препроцессинга на стороне клиента вынесено исходя из специфики работы алгоритма анализа DICOM-снимков, который реализован в рамках связной работы: метод, используемый в ней – свёрточная сеть для семантической сегментации, а именно – самая распространённая конфигурация полностью свёрточных сетей для семантической сегментации в разрезе медицинской сферы – U-Net [25].

Подобный метод анализа изображения ресурсоемок, поэтому целесообразно разделить задачу с помощью сервис-ориентированного подхода, разместив алгоритм анализа изображения на сервере, а на клиенте – предобработка перед анализом.

Клиентская часть ПО для предобработки выполняет следующие задачи:

- предобработка изображения;

- отправка массива данных на сервер;
- получение результата анализа с сервера.

По нажатию на кнопку «Open current image for preprocessing» в программе для просмотра DICOM-серий (рисунок 20), будут произведены подготовительные действия для создания задачи анализа (рисунок 21), а выбранное изображение будет открыто в окне для предобработки (рисунок 22).

Рисунок 20 – Открытие выбранного изображения в окне для предобработки

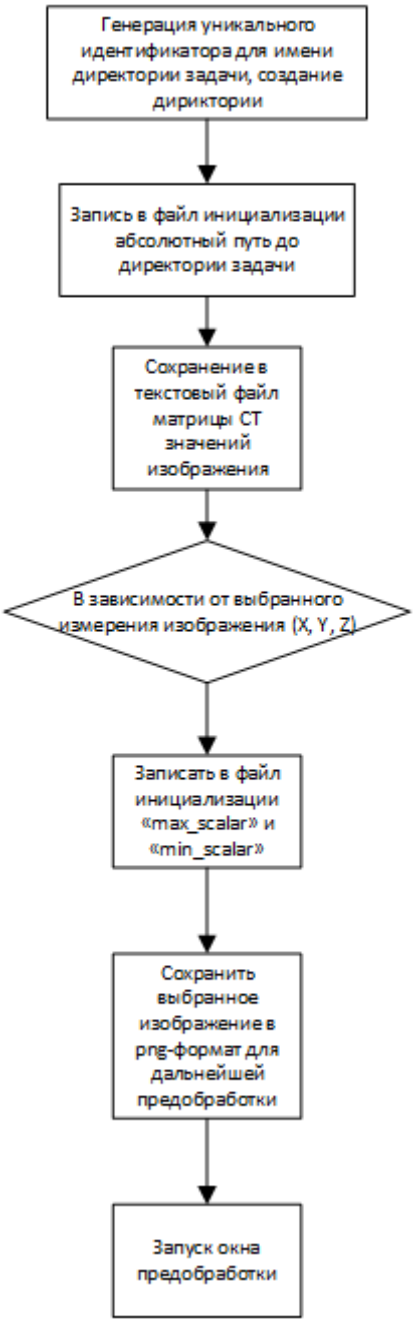

Рисунок 21 – Алгоритм создания задачи анализа выбранного изображения

Листинг данного алгоритма представлен в приложении В.

Рисунок 22 – Окно предобработки изображения

Для взаимодействия двух компонент клиентской части используются файлы инициализации с определенными в них строковыми параметрами, которые определяют текущие настройки ПО для предобработки (таблица 2).

Таблица 2 – Структура файла инициализации

Имя параметра	Описание
temp_folder	Абсолютный путь до директории, агрегирующей данные, необходимые для задач анализа изображений.
max_scalar	Максимальное значение плотности пикселя изображения величинах затухания радиоинтенсивности на исследуемых тканях [ссылка 32 из term 3].

Окончание таблицы 2

Имя параметра	Описание
min_scalar	Минимальное значение плотности пикселя изображения величинах затухания радиоинтенсивности на исследуемых тканях [ссылка 32 из term 3].
dicom_folder	Абсолютный путь до директории, содержащей серию DICOM-снимков в которую входит выбранное изображение.
curr_work_dir	Уникальный идентификатор директории, отвечающий за хранение данных по текущей задачи анализа.
api_url	URL, на котором развернута серверная часть ПО для анализа DICOM-серий.

Выбранное изображение должно быть подвергнуто нескольким алгоритмам предобработки – этого требует свёрточная нейронная сеть для анализа.

2.1.2 Исключение выбросных значений цветовой интенсивности изображения

Так как по шкале Хаунсфилда плотность костей измеряется от 400 единиц и выше, имеется также более подробная классификация шкалы для разных видов костей. Например, для значения 3000 соответствует пирамида височной кости, в то время как 1100 — кортикальная кость (например, челюсть). Так как в качестве входных данных всегда принимается грудная клетка, будет целесообразно избавиться от плотностей выше порога в 1200 единиц, так как информация о высокоплотных тканях, во-первых, избыточна, во-вторых, может по-

мешать при нормализации изображения (пиксели, отражающие мягкие ткани будут стремиться к нулю, а значит и плохо различимыми для нейронной сети) [26]. Поэтому здесь пиксели, имеющие значение числа Хаунсфилда более чем 1200 единиц, заменяются на чёрные пиксели со значением 0 (листинг 5).

Листинг 5 – Исключение значений цветовой интенсивности

```
1    for (int i = 0; i < image.Width; i++)
2    {
3            for (int j = 0; j < image.Height; j++)
4            {
5                if (CTvaluesOfImage[i, j] > 1200)
6                {
7                    result.SetPixel(i, j,
8                           Color.FromArgb(0, 0, 0));
9                 }
10       }
11   }
```

Данный алгоритм перебирает каждый пиксель сохраненного png-изображения, устанавливая значение 0 для тех областей снимка, CT-значение которых превышает 1200. Данные CT-значений (массив CTvaluesOfImage) получены из ранее сохраненной информации в текстовый файл.

2.1.3 Нормализация по шкале Хаунсфилда

Данное действие продолжает процесс исключения выбросных значений на изображении. В исходной версии кода производилось жёсткое обрезание границ Хаунсфилда от -100 до 400. Очевидно, авторы работ проводили прогнозирование на снимках, которые отправлялись с чётко определённого оборудования. В случае поставленной задачи оборудование может быть любым. Следовательно, необходимо выбрать такое сочетание нижней и верхней границы шкалы Хаунсфилда, чтобы на медицинском снимке были ясно различимы мягкие ткани. Затем, после того как пользователь устанавливает приемлемый интервал плотности тканей (рисунок 23), производится непосредственно нормализация – любые значения, выходящие за указанные границы, приравниваются к ним (листинг 6).

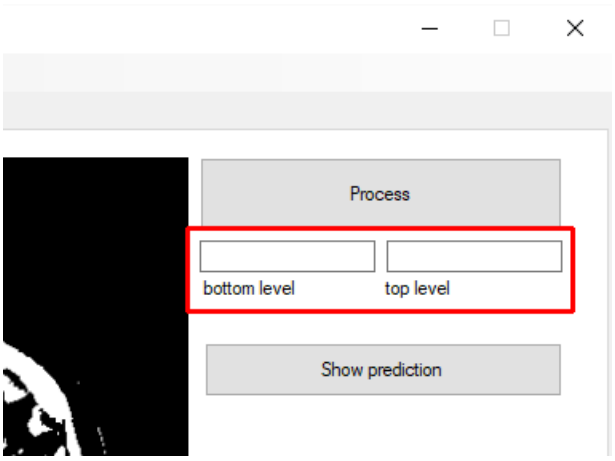

Рисунок 23 – Элементы управления для установки интервала плотностей тканей

Листинг 6 – Нормализация по шкале Хаунсфилда

```
1   for (int i = 0; i < image.Width; i++)
2   {
3       for (int j = 0; j < image.Height; j++)
4       {
5          if (CTvaluesOfImage[i, j] > houBorders[1])
6          {
7             result.SetPixel(i, j, Color.FromArgb(
8       houUpperToPixel, houUpperToPixel, houUpperToPixel));
9          }
10         else if (CTvaluesOfImage[i, j] < houBorders[0])
11         {
12            result.SetPixel(i, j, Color.FromArgb(
13                                        0, 0, 0));
14          }
15          Else
16          {
17             result.SetPixel(i, j,
18             Color.FromArgb(image.GetPixel(i, j).R,
19             image.GetPixel(i, j).R,
20             image.GetPixel(i, j).R));
21           }
22      }
23  }
```

Данный алгоритм реализует описанные выше действия по исключению выбросных значений. В строках 5 и 10 сравнивается CT-значение каждого пикселя изображения. В случае превышения верхней заданной границы плотности ткани, цвет пикселя установится в значение, соответствующее интенсивности черного цвета в зависимости от данного значения верхней границы (строки с 7

по 9). Преобразование значений Хаунсфилда в интенсивность четного цвета производится по формуле [27]:

$$HU = pixel\_value * slope + intercept, \quad (1)$$

где:

HU – значение пикселя по шкале Хаунсфилда,

pixel_value – значение пикселя представленного CT value,

slope, intercept – теги, содержащиеся в DICOM-файле, которые определяют линейное преобразование из пикселей (значений, которые представлены в памяти на жестком диске) в их представление в памяти [28].

В случае, если CT-значение пикселя ниже левой заданной границы плотности ткани, данное значение отбрасывается, цвет пикселя устанавливается в черный (строки с 12 по 13).

Если CT-значение попадает в заданный интервал (строка 15), цвет пикселя остается неизменным (строки с 17 по 20).

2.1.4 Шкалирование от 0 до 1

Обычно при работе с нейронными сетями данные масштабируются, или нормализуются относительно диапазона своих значений. Это необходимо для обеспечения одинаковой чувствительности нейронной сети по отношению к входящим признакам, так как минимизация целевой функции сети — алгоритм метрический, и может игнорировать близкие к нулю значения, что в итоге влияет на качество конечного результата: отсутствие нормальности остатков, низкая точность прогноза, попадание в локальные минимумы и т. д. (листинг 7).

Листинг 7 – Масштабирование значений интенсивности черного цвета

```
1   using (StreamWriter StrW =
2          new StreamWriter(TempDir + "\\normalize.txt"))
3   {
4      for (int i = 0; i < bmp.Width; i++)
5      {
6         for (int j = 0; j < bmp.Height; j++)
7         {
8            var pix = bmp.GetPixel(j, i);
```

Окончание листинга 7

```
9               double NormPixel = (double)(pix.R) / 255;
10              arrNormalizeImage.Add(NormPixel);
11              StrW.Write(Convert.ToString(NormPixel,
12                  System.Globalization.CultureInfo.
13                  InvariantCulture) + " ");
14          }
15          StrW.Write("\n");
16      }
17  }
```

Для шкалирования значений цвета выбранного изображения используется формула:

$$x_i' = \frac{x_i - \min(X)}{\max(X) - \min(X)}, \quad (2)$$

где x_i' – шкалированное значение пикселя;

x_i – текущее значение пикселя;

$\min(X)$ – минимальное значение пикселя текущего изображения;

$\min(X)$ – максимальное значение пикселя текущего изображения.

Так как обрабатываемое изображение представляет собой набор пикслей в градации серого, то формула может быть упрощена до:

$$x_i' = \frac{x_i}{255}, \quad (3)$$

где минимальное значение интенсивности пикселя – 0, а максимальное 255 (строка 9).

2.1.5 Понижение размерности

Нейронная сеть принимает входные данные строго определённой размерности. Существует множество методов для уменьшения размеров картинок. Здесь применяется один из наиболее простых алгоритмов — метод ближайшего соседа (рисунок 24). Для каждого пикселя конечного изображения выбирается один пиксель исходного, наиболее близкий к его положению с учетом масштабирования. Так, в данном случае размер снижается до 388 на 388 пикселей (листинг 8).

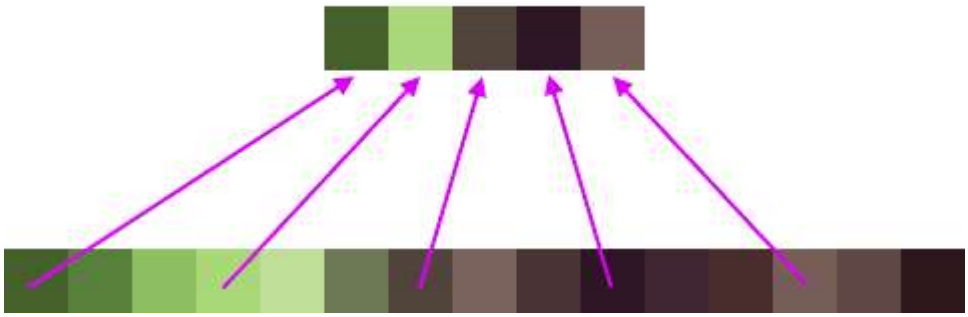

Рисунок 24 – Пример сжатия вектора пикселей
методом ближайшего соседа

Листинг 8 – Понижение размерности изображения

```
1   Bitmap result = new Bitmap(width, height);
2      using (Graphics g =
3             Graphics.FromImage(result))
4      {
5          g.InterpolationMode =
6             InterpolationMode.NearestNeighbor;
7          g.DrawImage(sourceBMP, 0, 0, width, height);
8      }
9   return result;
```

В ПО понижение размерности методом ближайшего соседа осуществляется стандартными средствами .Net, в пространстве имен InterpolationMode которого определены алгоритмы изменения размера изображения (строки с 5 по 7).

2.1.6 Паддинг

Паддинг — процесс обрамления изображения. Данная операция необходима, так как позволяет учитывать специфику свёрточных сетей, которые после прохождения по изображению ядром свёртки «зажёвывают» края снимка, тем самым утрачивая ценную для прогноза информацию. Потому применяется зеркальный паддинг — наложение на снимок фрагментов самого себя, начиная от края и до указанной границы (рисунок 25). После паддинга размер снимка эквивалентен 572 на 572 пикселей (листинг 9).

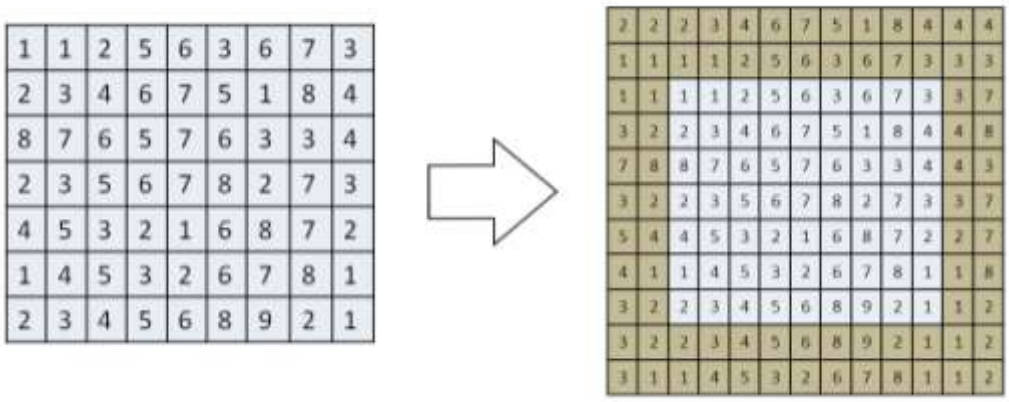

Рисунок 25 – Пример зеркального паддинга матрицы

Листинг 9 – Паддинг верхней границы изображения

```
1   for (int i = 0; i < image.Width; i++)
2   {
3      for (int j = 0; j < reflectParam / 2; j++)
4      {
5         resBtm.SetPixel(i + reflectParam / 2,
6                         reflectParam / 2 - j,
7
8                  Color.FromArgb(
9                  Convert.ToInt32(image.GetPixel(i, j).R),
10                 Convert.ToInt32(image.GetPixel(i, j).R),
11                 Convert.ToInt32(image.GetPixel(i, j).R)
12        ));
13     }
14  }
```

Аналогично добавляются значения интенсивности черного цвета по остальным краям выбранного изображения. Подробный листинг алгоритма паддинга представлен в приложении Г.

После всех операций предобработки выбранное изображение примет вид (рисунок 26):

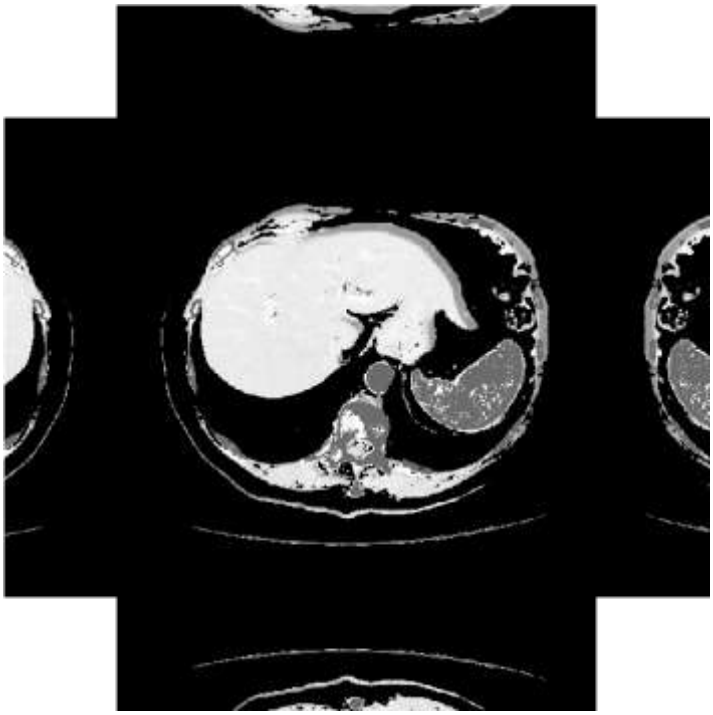

Рисунок 26 – Предобработанное изображение

2.1.1 Отправка данных на сервер

После операций предобработки текущее окно примет вид (рисунок 27).

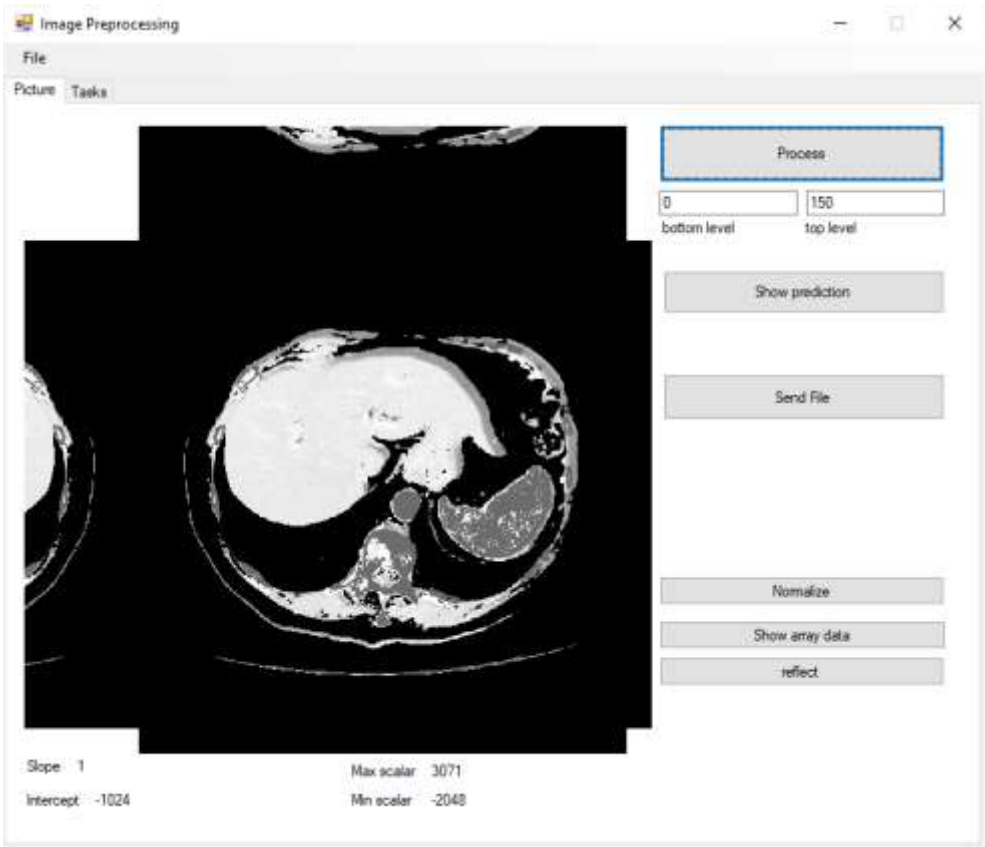

Рисунок 27 – Конец операций предобработки

Для отправки текстового файла, содержащего информацию для алгоритма анализа, используется метод, вызываемый по нажатию на кнопку «Send file» (листинг 10).

Листинг 10 – Обработка ответа от сервера

```
1   System.IO.Stream response = Upload(ApiURL +
2      "application/set-task-in-db",
3      CurrentHash,
4      File.Open(TempDir + @"\normalize.txt", FileMode.Open),
5      "task_hash",
6      "file_1",
7      "normalize.txt");
8   if (response != null)
9   {
10     using (var reader =
11             new StreamReader(response, Encoding.UTF8))
12     {
13        string value = reader.ReadToEnd();
14        Newtonsoft.Json.Linq.JToken token =
15           Newtonsoft.Json.Linq.JObject.Parse(value);
16        MessageBox.Show("Message from server: " +
17          (string)token.SelectToken("message"));
18     }
19  }
20  else
21  {
22     MessageBox.Show("File do not send!");
23  }
```

В строках с 1 по 7 вызывается метод отправки информации о предобработанном изображении на сервер, в который в качестве параметров передаются:

- Имя метода на сервере для приема информации (строка 2);
- Текущее хэш-имя директории задачи (строка 3);
- Поток на чтение файла, требующего отправки на сервер (строка 4);
- Имя параметра POST-запроса, содержащего хэш-имя директории задачи (строка 5);
- Имя параметра POST-запроса, содержащего отправляемый файл (строка 6);
- Имя файла, требующего отправки на сервер (строка 7).

Если от сервера будет получен ответ (строка 8), то используя объект .Net StreamReader, будет прочитан поток передаваемого сообщения в кодировке UTF-8 (строки 10 по 11).

Так как ответы от сервера передаются в формате JSON, то с помощью библиотеки Newtonsoft.Json будет получено объектно-ориентированное представление полученной JSON-строки (строки с 14 по 15), и выведено на экран сообщение, содержащееся по ключу «message» (строки с 16 по 17).

Если ответ от сервера не был получен, то на экран будет выведено сообщение содержащее ошибку соединения.

2.1.2 Получение результата от сервера

Результат работы алгоритма анализа DICOM-снимка есть файл, содержащий матрицу-маску для выбранного изображения, где бинарно помечен каждый пиксель снимка: 1 – если нейросеть считает, что в данном месте искомый орган, 0 – если нет. Данный файл помещается на сервере в директорию задачи, из которой может быть получен клиентом.

Для просмотра всех задач и выбора снимка для демонстрации прогноза предусмотрена вкладка «Tasks» в окне предобработки (рисунок 28).

Файл прогноза может быть получен при выборе строки в элементе DataGridView соответствующей одной задаче (листинг 11).

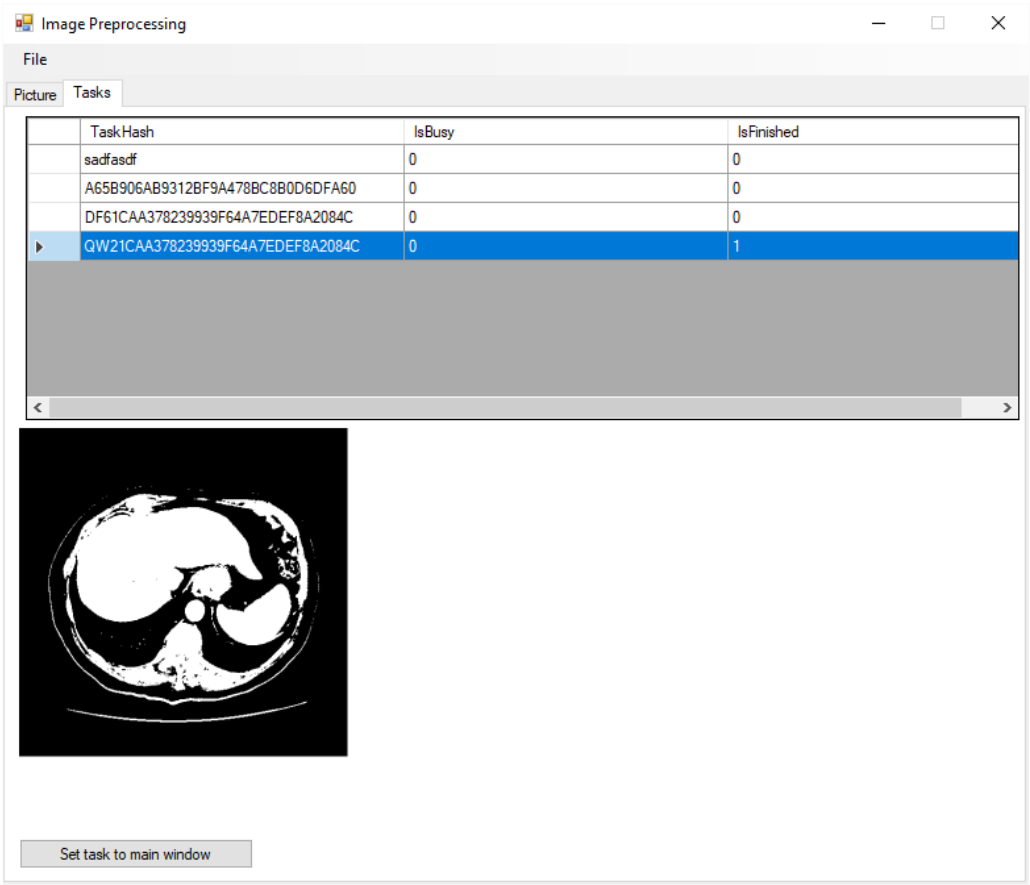

Рисунок 28 – Список задач

Листинг 11 – Выбор задачи

```
1   OpenTask.Enabled = false;
2   if (dataGridView1.SelectedRows.Count == 1)
3   {
4       string pathToImage =
5       IniObjMain.IniReadValue("Info", "temp_folder") + "\\"
6       + dataGridView1.SelectedRows[0].Cells[0].
7       Value.ToString() + @"\image.png";
8       if (File.Exists(pathToImage))
9       {
10          TaskPagePicturePictureBox.Image =
11          Image.FromFile(pathToImage);
12      }
13      else
14      {
15          TaskPagePicturePictureBox.Image =
16          TaskPagePicturePictureBox.InitialImage;
17      }
18      if(dataGridView1.SelectedRows[0].Cells[1].Value.
19         ToString() == "0" &&
20         dataGridView1.SelectedRows[0].Cells[2].Value.
21         ToString() == "1")
22      {
```

Окончание листинга 11

```
23          using (HttpClient httpClient = new HttpClient())
24          {
25              try
26              {
27                string task_hash =
28                dataGridView1.SelectedRows[0].Cells[0].
29                Value.ToString();
30                var uri = ApiURL +
31                "application/file-prediction-ready?hash=" +
32                task_hash;
33                var response =
34                await httpClient.GetAsync(uri);
35                string JsonFromResponse =
36                await response.Content.
37                ReadAsStringAsync();
38                Newtonsoft.Json.Linq.JToken token =
39                Newtonsoft.Json.Linq.JObject.
40                Parse(JsonFromResponse);
41                string state_of_response =
42                (string)token.SelectToken("state");
43                if(state_of_response == "1")
44                {
45                 using (var client =
46                        new System.Net.WebClient())
47                 {
48                  client.DownloadFile(ApiURL +
49                  "/" + task_hash + "/pred.txt",
50                  IniObjMain.IniReadValue("Info",
51                  "temp_folder") + "\\" +
52                  task_hash + "\\pred.txt");
53                 }
54                 OpenTask.Enabled = true;
55                }
56                else
57                {
58                 OpenTask.Enabled = false;
59                 MessageBox.Show("Message from server: "
60                 + (string)token.SelectToken("message"));
61                }
62               }
63               catch (Exception ec)
64               {
65                MessageBox.Show(ec.Message,
66                  "Connection error",
67                  MessageBoxButtons.OK,
68                  MessageBoxIcon.Stop);
69               }
70           }
71      }}
```

В строке 1 кнопка, открывающая выбранную задачу, на время проверки доступности ответа от алгоритма анализа становится неактивной.

Со строки 4 по строку 11 проверяется доступен ли предпоказ изображения для данной задачи: если данное изображение имеется в директории задачи, оно будет выведено в окно просмотра на вкладке «Tasks» (рисунок 28). Иначе, в окно предпросмотра будет выведено стандартное изображение (строки с 15 по 16).

Далее, производится проверка флагов задачи для определения завершенности анализа (строки с 18 по 21). Если флаг IsBusy установлен в значение 0 (задача не занята алгоритмом анализа), а флаг IsFinished установлен в 1 (алгоритм анализа закночил свою работу), то станет возможным скачать результат.

Перед получением файла с матрицей-маской, отправится запрос на сервер, для определения существует ли требуемый файл (строки с 27 по 42). Если ответ от сервера по ключу «state» будет равен 1 – файл может быть скачен. Иначе, будет выведено в информационном окне сообщение об ошибке от сервера (сроки с 59 по 60).

Файл будет скачен с сервера из директории с хэш-именем задачи в соответствующий каталог клиента (строки с 48 по 52).

Если во время соединения возникнет необработанное исключение (например, разрыв соединения), ошибка будет выведена в информационном окне (строки с 63 по 50).

После получения файла, кнопка загрузки выбранной задачи станет активна (строка 54).

2.1.3 Просмотр результата анализа

Просмотр результата анализа заключается в наложении матрицы-маски на исходное изображение, закрашивая зелёным цветом пиксели, которым соответствует значение 1 маски.

Загрузка данных задачи осуществляется нажатием на кнопку «Set task to main window» (листинг 12).

Листинг 12 – Загрузка данных задачи

```
1   CurrentHash =
2   dataGridView1.SelectedRows[0].Cells[0].Value.ToString();
3   IniObjMain.IniWriteValue("Info", "curr_work_dir",
4      CurrentHash);
5   TempDir = IniObjMain.IniReadValue("Info", "temp_folder")
6      + "\\" + CurrentHash;
7   pictureBox1.Image =
8      Image.FromFile(TempDir + @"\image.png");
9   MessageBox.Show("Image has been loaded on first tab"
10     + "\nShowing prediction is available");
```

В строке 1 в переменную CurrentHash записывается хэш-имя текущей задачи, в соответствии с выбранной строкой в элементе DataGridView.

Со строки 3 по 4 в файл инициализации устанавливается параметр, отвечающий за хэш-имя текущей директории задачи.

В строке 5 в переменную TempDir записывается абсолютный путь до каталога задачи.

Со строки 7 по строку 8 в главное окно просмотра изображения устанавливается ранее выбранное необработанное изображение данной задачи.

В конце проведенных операций будет выведено информационное окно, содержащее сообщение об окончании загрузки (строки с 9 по 10).

Отображение результатов анализа на вкладке «Picture» осуществляется нажатием на кнопку «Show prediction» (рисунок 29).

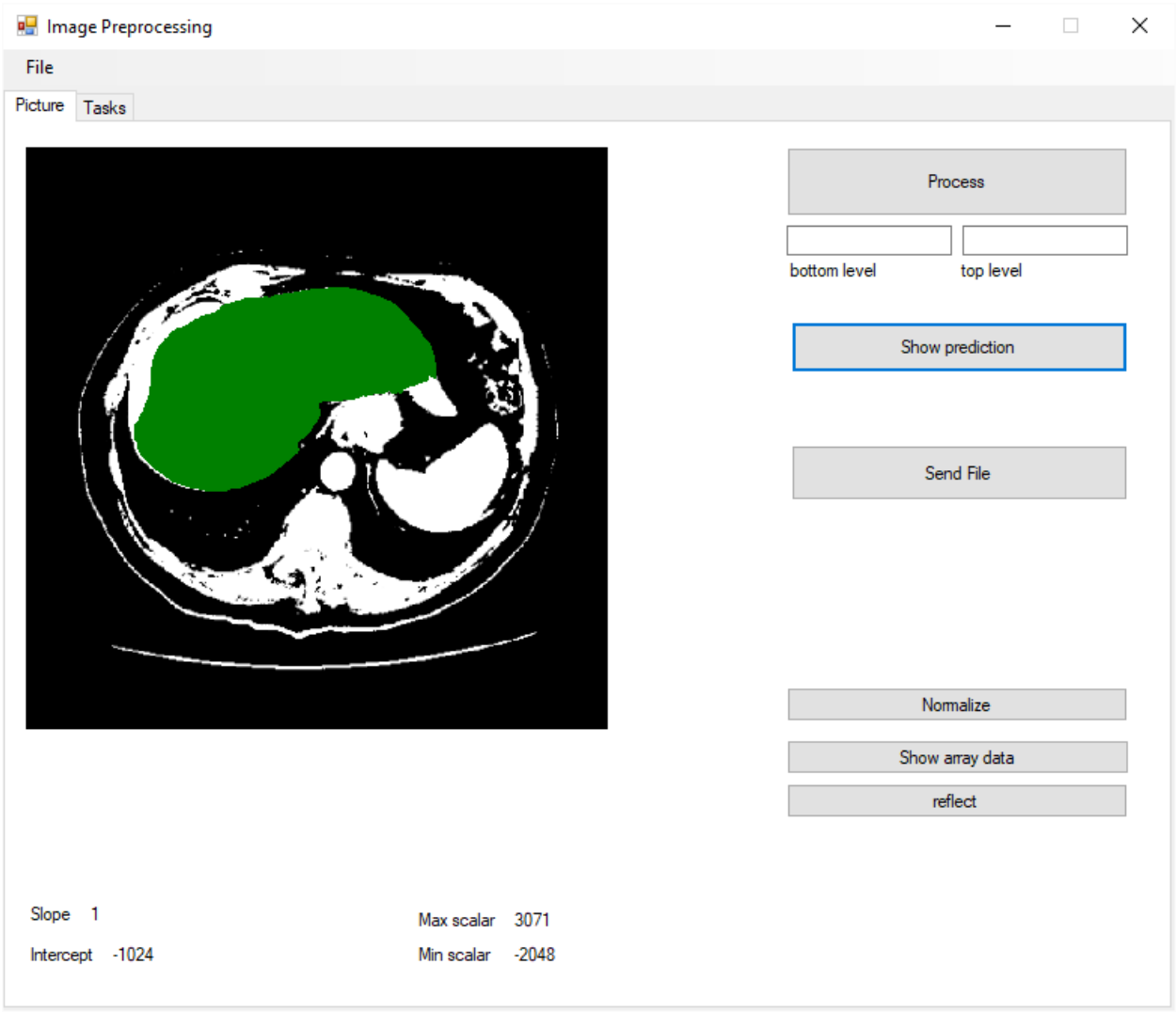

Рисунок 29 – Результат анализа выбранного DICOM-снимка

Выводы по главе 2

В главе 2 рассмотрены используемые средства разработки. Описана программная реализация клиентской части ПО для просмотра DICOM-снимков. Приведены алгоритмы и принципы работы с ПО.

3 РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Для реализации взаимодействия двух компонент проекта по анализу DICOM-снимков, будет целесообразно реализовать web-приложение, обеспечивающее передачу входных и выходных данных, а также обеспечивающее работу с базой данных для хранения записей о готовых и требующих анализа задачах.

3.1 Обзор используемых средств разработки

3.1.1 Zend Framework

Для реализации серверной и web частей проекта, лучше всего использовать технологии, которые дают возможность избежать часто встречающиеся издержки при разработке web-приложений. Такие технологии носят название «фреймворк» (framework) — программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта [29].

Для решения задачи реализации web части проекта, описываемого в данной работе, будет использован Zend Framework.

Zend Framework — свободный фреймворк на PHP для разработки веб-приложений, разрабатываемый компанией Zend.

Основывается на принципах MVC. Помимо MVC-компонентов содержит множество библиотек, полезных для построения приложения, например, реализованы компоненты для интеграции со сторонними сервисами — YouTube, del.icio.us и другими. Начиная с версии 1.6 поставляется с JavaScript-фреймворком Dojo, а также включает в себя компоненты для работы с ним. В сентябре 2012 года вышла версия 2.0 (Zend Framework 2). В июне 2016 года вышла версия 3.0 (Zend Framework 3) [30].

3.1.2 LAMP

Для обеспечения работы web-приложения используется широко популярный набор серверного программного обеспечения, а именно операционная система Ubuntu Server 16.04, веб-сервер Apache 2, база данных под управлением СУБД MySql и язык web-программирования PHP. Данный набор инструментов обусловлен, в первую очередь, свободой распространения, а также удобством пользования и легкостью обслуживания.

3.1.3 Краткий принцип работы Zend-приложения

Начиная с запроса браузером информации от веб-сервера по адресу URL, до получения браузером HTTP ответа, принцип работы web-приложения можно описать следующими шагами (рисунок 30):

1. Веб сервер Apache 2 по стандартному порту 80 получает HTTP запрос на получения информации от соответствующего URL адресу контроллера Zend web-приложения. Zend приложение для web-сервера предоставляет одну точку входа – файл index.php, что позволяет скрыть явное отображение к исходным кодам программы.
2. С помощью опции mod_rewrite веб сервера и файлам конфигурации Zend Framework, запрос получает определенный метод, реализующий необходимую логику на языке PHP.
3. Если метод отработает без ошибок, сформируется объект класса ViewModel, содержащий в себе результат работы кода PHP и содержимое Web-страницы на стандартизированном языке разметки HTML.
4. Данная разметка будет отдана web-серверу для формирования ответа браузеру.
5. Браузер отобразит полученную Web-страницу.

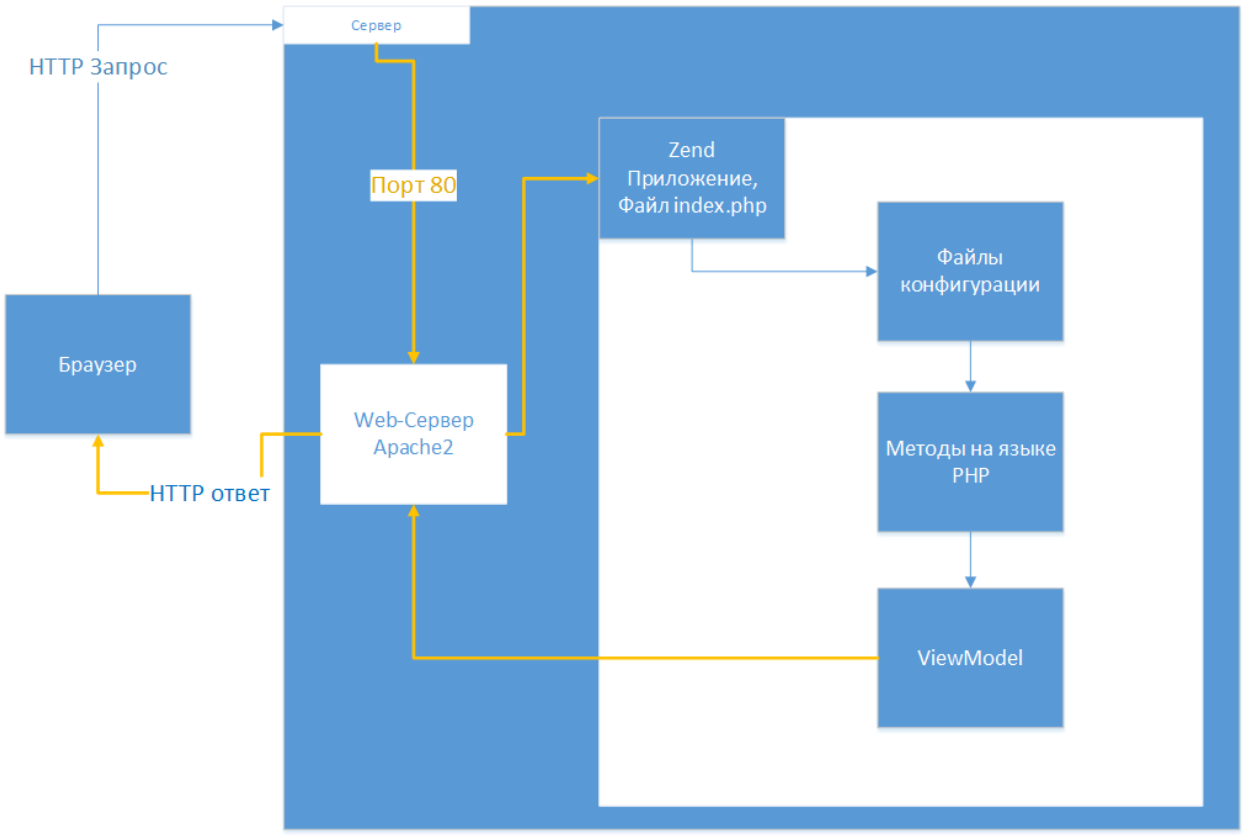

Рисунок 30 – Принцип работы web-приложения

3.2 Реализация серверной части программного обеспечения

3.2.1 Установка LAMP

Для разработки и отладки web-сервера вполне достаточно будет не иметь полноценный физический компьютер – на его месте будет виртуальная машина.

В качестве среды виртуализации будет использовать бесплатное ПО VirtualBox (рисунок 31).

В качестве виртуальной машины установим Ubuntu 16.04. Официальный дистрибутив Ubuntu бесплатен и скачивается с сайта проекта http://releases.ubuntu.com/16.04/.

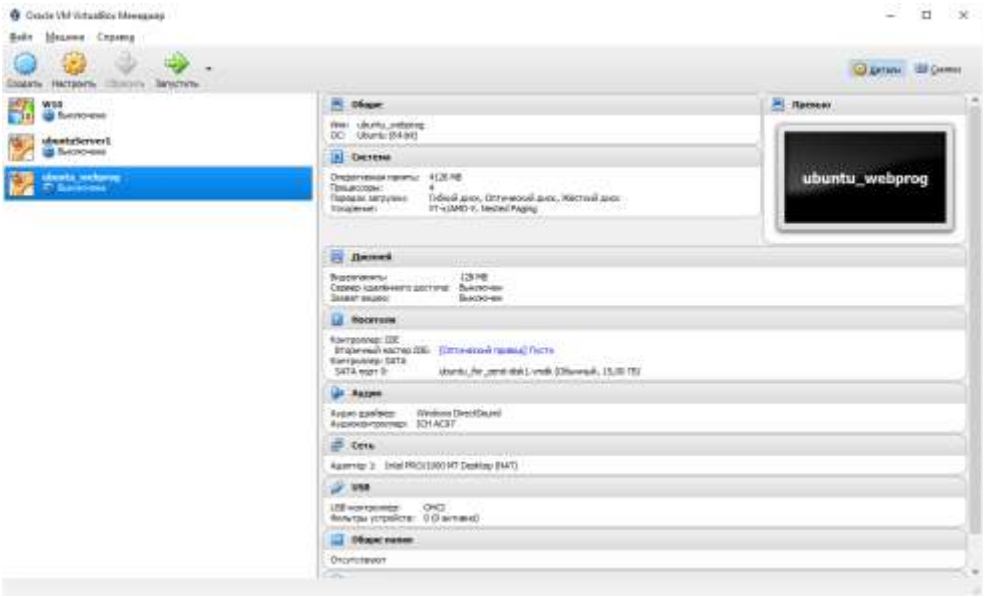

Рисунок 31 – Рабочее окно VirtualBox

Виртуальная машина для реализуемого ПО требует следующие предустановки:

- на вкладке «Носители» укажем образ Ubuntu 16.04, скаченный с официального сайта (рисунок 32);
- на вкладке «Система» укажем желаемое количество оперативной памяти, выделяемой под виртуальную машину, а также количество процессоров для работы с ОС (рисунок 34);
- для того, чтобы виртуальная машина была видна в локальной сети, на вкладке «Сеть» определим тип подключения сетевого адаптера как «Сетевой мост» (рисунок 33);

Для получения стека LAMP, после установки, выполним в терминале Ubuntu необходимые команды для установки ПО:

- команда для установки Apache 2 (листинг 13);
- команда для установки MySql (листинг 14);
- команда для установки php (листинг 15);

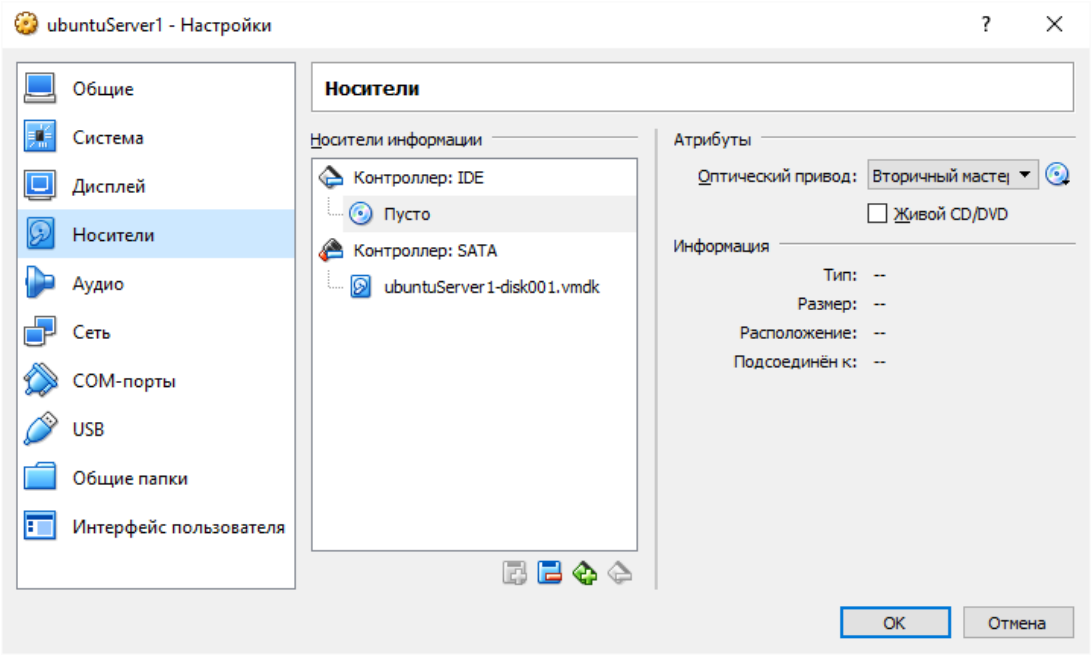

Рисунок 32 – Вкладка «Носители» для указания образа ОС

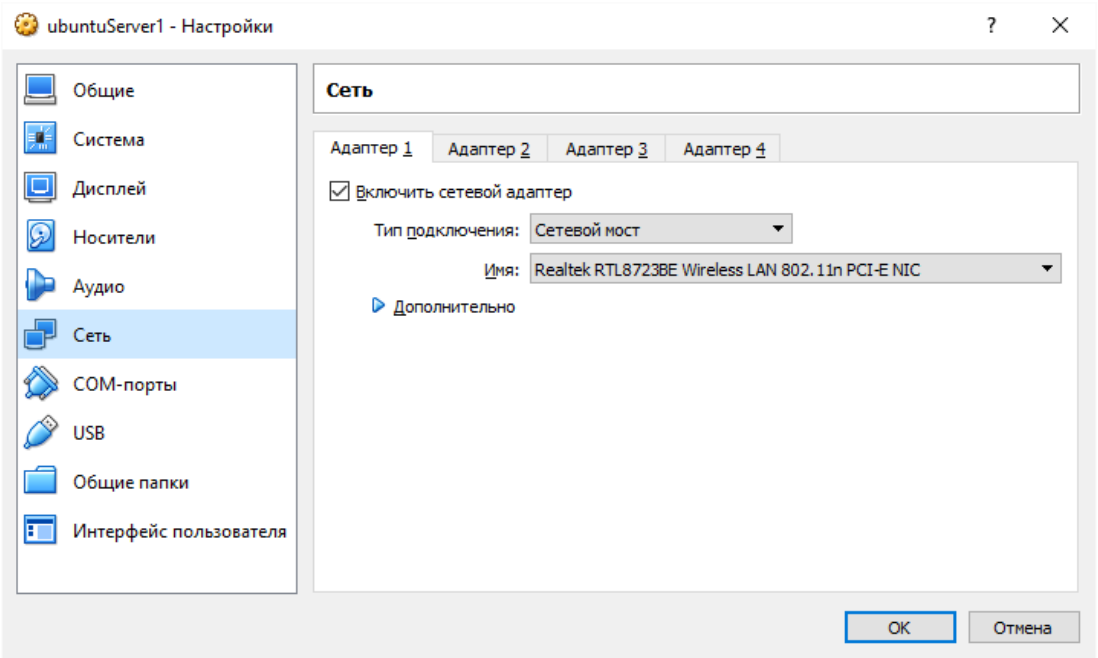

Рисунок 33 – Настройка сети виртуальной машины

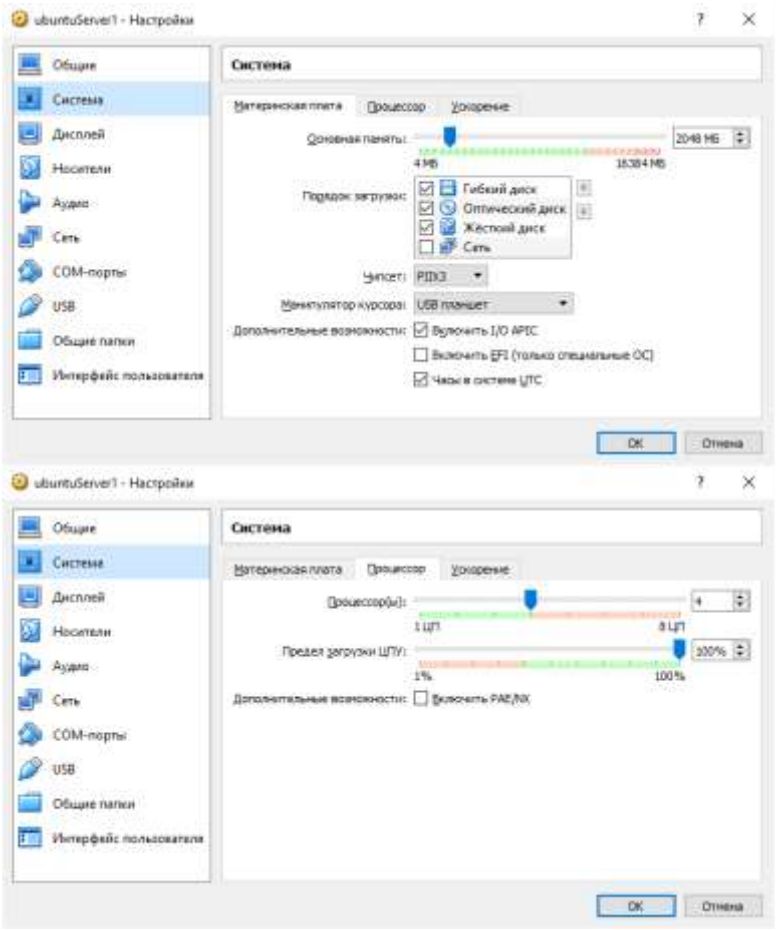

Рисунок 34 – Настройка RAM и ЦП

Листинг 13 – Команда для установки Apache 2

```
1   sudo apt-get install apache2
```

Листинг 14 – Команда для установки MySql

```
1   sudo apt-get install mysql-server mysql-client
```

Листинг 15 – Команда для установки php

```
1   sudo apt-get install php7.0-mysql php7.0-curl
2   php7.0-json php7.0-cgi  php7.0 libapache2-mod-php7.0
3   php-mbstring php7.0-mbstring php-gettext
```

После установки Apache2 и MySql, чтобы убедиться, что необходимые сервисы запущены, выполним соответствующие команды (листинг 16). Сервисы ПО отобразят свой статус в терминале вывода (рисунок 35).

Листинг 16 – Команды для проверки активности Apache2 и MySql

```
1   service apache2 status
2   service mysql status
```

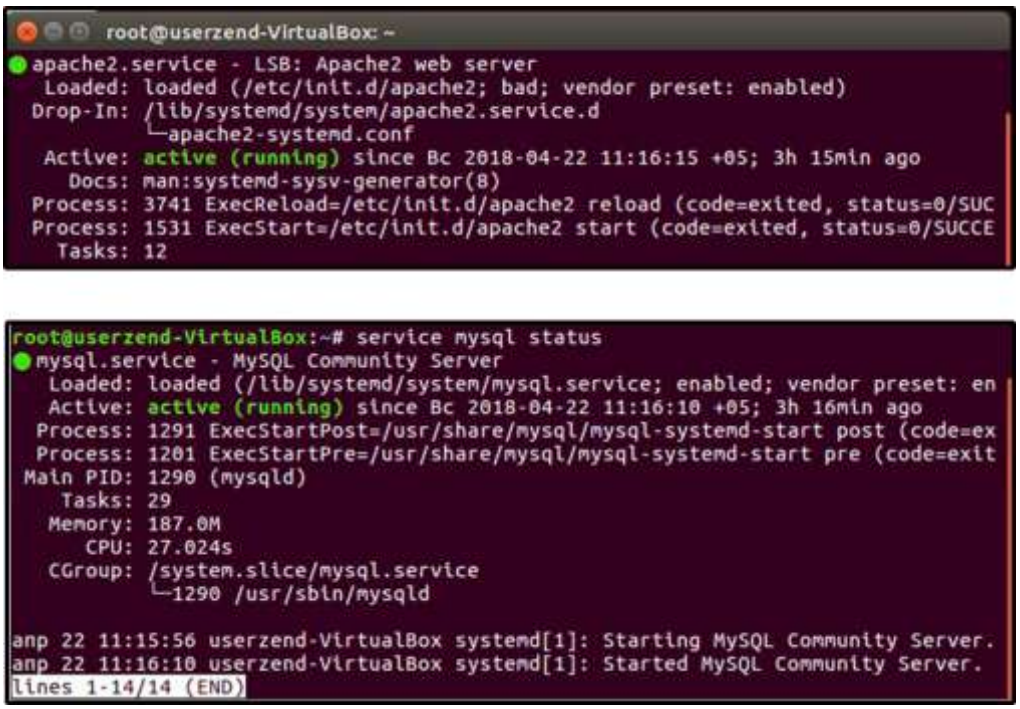

Рисунок 35 – Ответ «active» (активен) от сервисов Apache2 и MySql

3.2.2 Установка и настройка Zend Framework

Чтобы начать разрабатывать web-приложение с применением Zend Framework, достаточно скачать так называемое «скелетное приложение» (СП) – это простой вебсайт на основе ZF3, который содержит основные необходимые вещи для создания собственных приложений.

СП устанавливается с помощью Composer – это пакетный менеджер уровня приложений для языка программирования PHP, который предоставляет средства по управлению зависимостями в PHP-приложении [31] – поэтому, сначала требуется установить его (листинг 17).

Листинг 17 – Команды для установки Composer

```
1   php     -r     "copy('https://getcomposer.org/installer',
    'composer-setup.php');"
2   php composer-setup.php
```

В строке 1 вызывается интерпретатор php, который исполняет строку на этом языке заключенную в двойные кавычки, а именно, происходит копирование php файла «composer-setup.php» с указанного URL-адреса. Файл «composer-

setup.php» содержит в себе инструкции на языке php, которые позволят установить Composer.

В строке 2 вызывается интерпретатор php и исполняется скаченные файл «composer-setup.php».

После установки Composer станет возможным скачать скелетное приложение Zend Framwork (листинг 18).

Листинг 18 – Команда для скачивания скелетного приложения

```
1   php composer.phar create-project –sdev
2   zendframework/skeleton-application helloworld
```

3.2.1 Настройка web-сервера Apache2

Чтобы web-приложение отвечало на http запросы от браузера или от внешних сервисов, необходимо настроить сервер Apache2. Для этого в файле конфигурации укажем путь до директории приложения (листинг 19).

Листинг 19 – Файл конфигурации Apache2

```
1   <VirtualHost *:80>
2   ServerAdmin webmaster@localhost
3   DocumentRoot /var/www/zendserver/public
4   <Directory /var/www/zendserver/public/>
5   DirectoryIndex index.php
6   AllowOverride All
7   Require all granted
8   </Directory>
9   </VirtualHost>
```

В строках 3 и 4 указаны настройки исходной директории, где располагается web- приложение. В строке 5 указан исполняемый php-файл – точка входа в приложения.

После настройки сервера, если все сделано верно, при обращении через браузер к локальному ip-адресу виртуальной машины, станет доступна страница приветствия скелетного приложения Zend Framework (рисунок 36).

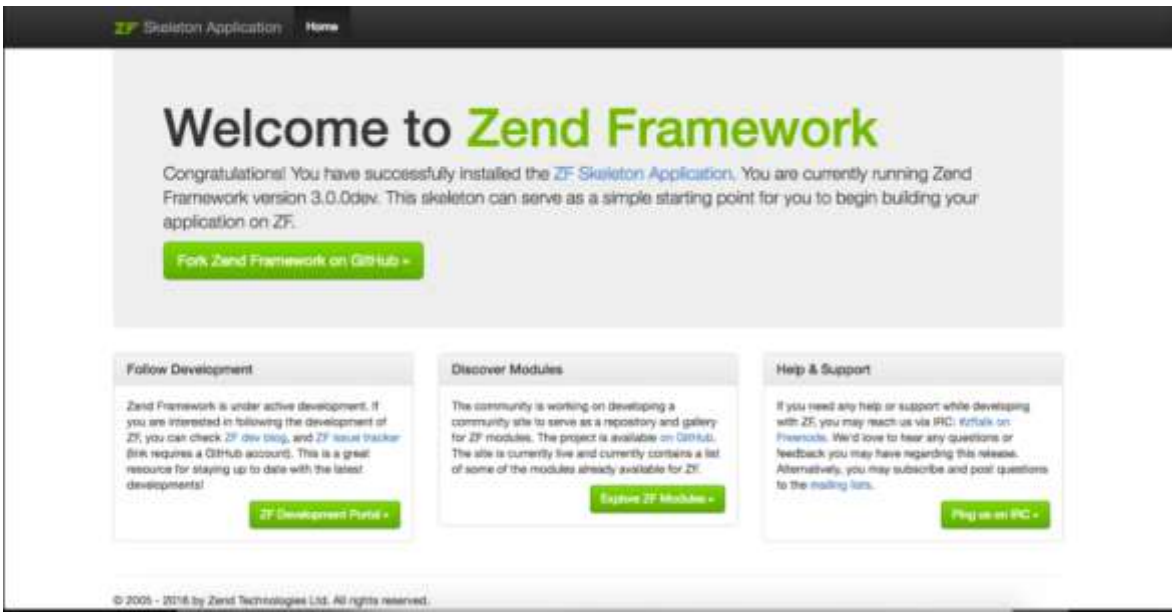

Рисунок 36 – Приветственная страница скелетного приложения Zend Framework 3

3.2.1 Структура проекта Zend Framework

Zend Framework реализован с помощью паттерна MVC. Он разделяет работу своих компонент на три отдельные функциональные роли: модель, представление и контроллер. В данном паттерне модель не зависит от представления или управляющей логики, что делает возможным проектирование модели как независимого компонента и, например, создавать несколько представлений для одной модели [32].

Основной каталог проекта имеет следующие подкаталоги:

- каталог «config» содержит файлы конфигурации приложения;
- каталог «data» содержит данные, которые может создать web-приложение;
- каталог «module» содержит все модули приложения, которые реализуют логику работы web-приложения;
- каталог «vendor» содержит файлы сторонних библиотек;
- каталог «public» содержит данные, которые будут доступны для внешних пользователей web-приложения.

По-умолчанию в «module» создан один модуль – «Application», доступный для редактирования и добавления своих правил работы web-приложения.

В подкаталоге «src» содержатся классы-контроллеры web-приложения – именно здесь будет реализована основная работы web-API.

В подкаталоге «view» содержатся файлы-представления на языке гипертекстовой разметки html. Так как реализуемое web-приложение несёт в себе функциональность web-API, нет необходимости реализовывать на каждый запрос к методам класса-контроллера соответствующее представление.

Чтобы определить модели данных, используемых web-приложением, создается директория «Entity» рядом с директорией «Controller».

Результирующая структура основных директорий проекта Zend Framework представлена на рисунке (рисунок 37).

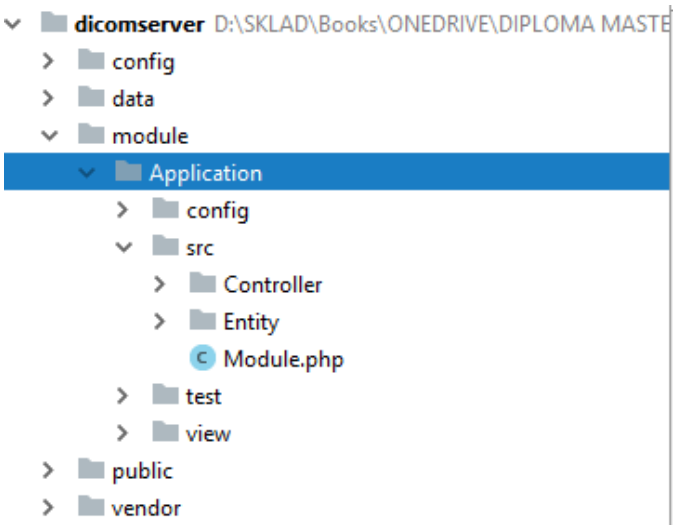

Рисунок 37 – Результирующая структура директорий

3.2.2 Реализация модели данных для web-приложения

Целью связывания работы программы для просмотра DICOM-снимков и алгоритма анализа с применением машинного обучения, написанного на языке python являются:

- передача информации о снимке, который должен быть проанализирован;

- передача готового результата анализа обратно приложению для просмотра DICOM-снимков.

Из поставленных целей, вытекают следующие задачи, решаемые организацией базы данных:

- создание сущности «Задача», для разделения разных снимков, требующих анализа;
- создание модели данных для сущности «Задача» для обеспечения работы web-API с БД;
- получение всех задач из БД по запросу программы для просмотра DICOM-снимков.

Для работы с БД в Zend Framework рекомендуется использовать объектно-реляционный проектор (ORM) Doctrine 2, который базируется на слое абстракции доступа к БД (DBAL). Одной из ключевых возможностей Doctrine является запись запросов к БД на собственном объектно-ориентированном диалекте SQL, называемом DQL (Doctrine Query Language) и базирующемся на идеях HQL (Hibernate Query Language) [33].

Doctrine 2 также поддерживает подход программирования «code first», это значит, что возможно сначала описать php-классы, определяющие модель сущностей БД и, в последствии, с помощью инструмента doctrine-module, создать необходимые таблицы в БД на основе описанных классов.

Для решения задачи определим таблицу «Tasks» с помощью соответствующего php-класса (листинг 20).

Листинг 20 – Класс-модель «Tasks»

```
1   /**
2    * @ORM\Entity
3    * @ORM\Table(name="tasks")
4    */
5   class Tasks
6   {
7       /**
8        * @ORM\Id
9        * @ORM\Column(type="integer")
```

Окончание листинга 20

```
10        * @ORM\GeneratedValue(strategy="AUTO")
11        */
12       protected $id;
13       /**
14        * @ORM\Column(type="string")
15        */
16       protected $task_hash;
17       /**
18        * @ORM\Column(type="integer")
19        */
20       protected $is_busy;
21       /**
22        * @ORM\Column(type="integer")
23        */
24       protected $is_finished;
25   }
```

В строках 1-4 определяется декоратор ORM, с помощью которого Doctrine определит какую таблицу БД описывает данный класс.

В строках 12, 16, 20 и 24 определены поля класса, которые декорируются по такому же принципу (например, в строках с 7 по 11), чтобы определить столбцы таблицы БД. Описание столбцов приведено в таблице (таблица 3).

Таблица 3 – Описание полей таблицы «Tasks»

Название поля	Описание
id	Уникальный идентификатор записи.
task_hash	Уникальный идентификатор созданной задачи, требующей анализа.
is_busy	Флаг, определяющий обрабатывается ли алгоритмом анализа данная задача.

Окончание таблицы 3

Название поля	Описание
is_finished	Флаг, определяющий закончена ли работа алгоритма анализа над этой задачей.

После описания класса-модели, выполним команду (листинг 21), чтобы Doctrine создала в подключенной к проекту БД соответствующую таблицу (рисунок 38).

Листинг 21 – Команда для создания таблиц в БД

```
1    ./doctrine-module orm:schema-tool:update --force
```

Рисунок 38 – Автоматически созданная таблица в БД MySql на основе php-класса

3.2.3 Реализация удаленных методов web-приложения

Взаимодействие между двумя компонентами проекта по просмотру и анализу DICOM-снимков будет производится по средствам вызова удаленных процедур. Все методы, отвечающие за работу с web-API располагаются в классе IndexController.php модуля «Application».

Механизм вызова метода web-API представляет собой GET и POST-запросы к URL-адресам методов в класса IndexController.php. Пример URL-адреса приведен в листинге (листинг 22).

Листинг 22 – Пример URL-адреса

```
1   192.168.1.6/application/set-ready-analysis
```

Запрос имеет следующую структуру:

- «**192.168.1.6**» определяет ip-адрес компьютера с включённым web-сервером для обеспечения доступа к web-API;
- «**application**» определяет модуль web-приложения Zend Framework;
- «**set-ready-analysis**» определяет имя вызываемой удаленной процедуры, реализованной в классе IndexController.php.

Для решения поставленных задач, определенных в разделе 3.2.2, в классе IndexController.php будут реализованы следующие методы:

- «**gettasksAction**» – метод для получения данных о всех задачах из БД;
- «**setTaskInDbAction**» – метод для постановки задачи, требующей анализа;
- «**setReadyAnalysisAction**» – метод для установки статуса «Готово» для задачи и получения прогноза от нейронной сети;
- «**filePredictionReadyAction**» – метод для установления наличия файла с результатами анализа задачи;
- «**consolestarttaskAction**» – метод запуска программы для анализа DICOM-снимка, запускается раз в 5 минут планировщиком заданий CRON, установленном на сервере.

Схема взаимодействия API с использующими его сервисами представлена на рисунке (рисунок 39).

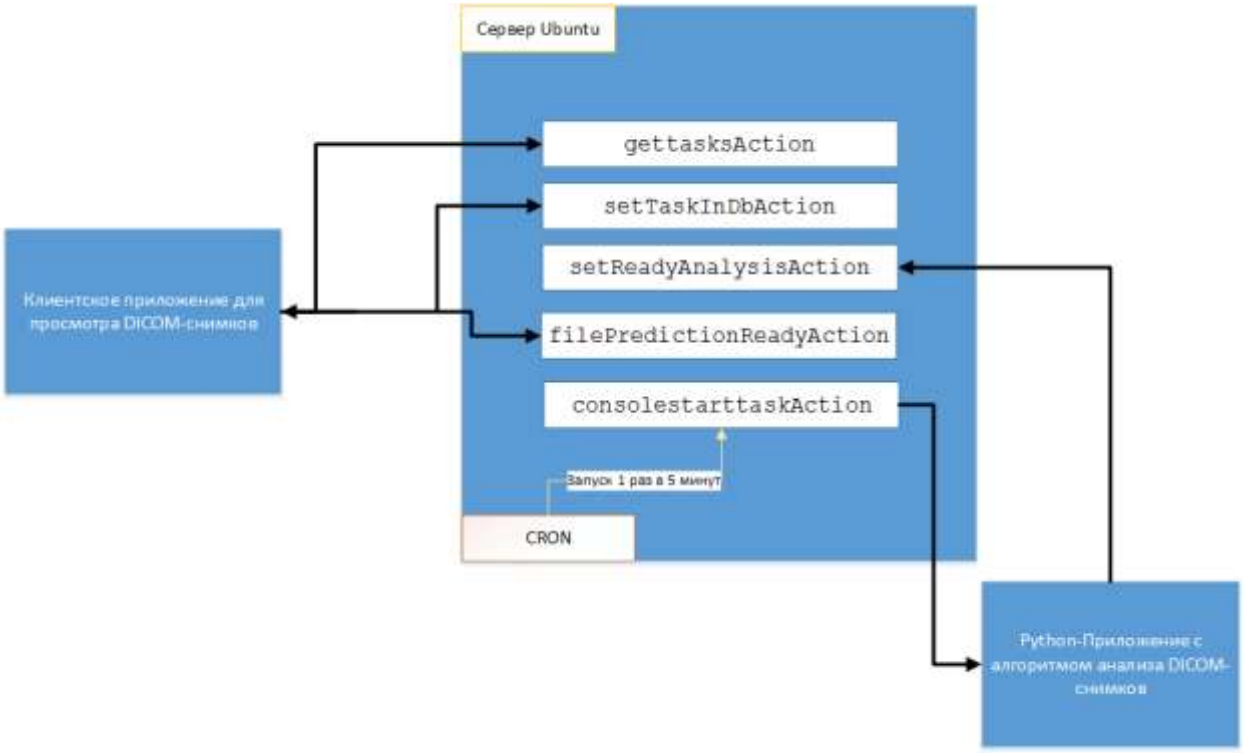

Рисунок 39 – Схема взаимодействия с API

Методы, реализованные на сервере, передают вызывающему коду ответ в формате JSON – простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером, основанный на подмножестве языка программирования JavaScript, определенного в стандарте ECMA-262 3rd Edition - December 1999 [34].

Пример ответа представлен на рисунке:

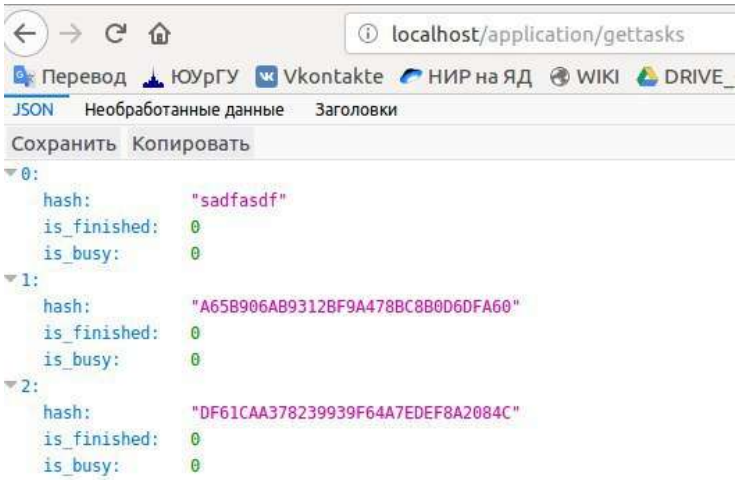

Рисунок 40 – Пример ответа от сервера в JSON-формате

Метод «gettasksAction» предоставляет доступ ко всем записям БД, тем самым программа для просмотра DICOM-снимков может получить полный список задач, которые в очереди или имеют статус «Завершен» (листинг 23).

Листинг 23 – Метод получения всех здачач из БД

```
1   public function gettasksAction(){
2           $arr = $this->
3                  ORM->getRepository
4                  ('\Application\Entity\Tasks')->findAll();
5           $arrRes = [];
6           foreach ($arr as $item){
7               $toResponse = [
8                   'hash' => $item->getTaskHash(),
9                   'is_finished' => $item->getIsFinished(),
10                  'is_busy' => $item->getIsBusy(),
11              ];
12              $arrRes[] = $toResponse;
13          }
14          return new JsonModel($arrRes);
15      }
```

В строке 1 определяется имя метода. К каждому методу, к которому планируется осуществлять доступ через URL, в конце, должен быть добавлен суффикс «**Action**», так как система роутинга (маршрутизации) Zend Framework требует этого.

В строках 3 по 4 осуществляется запрос через объектно-ориентированный проектор Doctrine 2 к таблице БД «**Tasks**» и запрашиваются все записи данной сущности. ORM возвращает ответ в виде стандартного php-массива, который помещается в переменную «**$arr**». Массив содержит в себе php-объекты класса «**Tasks**».

В строке 5 объявляется php-массив для последующего формирования ответа в формате JSON.

В строках с 6 по 13 в цикле по всем вхождениям в массиве с объектами типа «**Tasks**» формируется запись ответа и помещается в массив, определенный в строке 5. В ответ входят:

- уникальный хэш задачи (строка 8);
- флаг, показывающий законченная ли данная задача (строка 9);

- флаг, показывающий обрабатывается ли данная задача в настоящий момент (строка 10).

В строке 14, в качестве ответа от функции возвращается объект, реализованный в Zend Framework, позволяющий сформировать ответ от сервера в формате JSON на основе php-массива.

Метод «setTaskInDbAction» предназначен для отправки данных о задаче, требующей анализа, от клиента для просмотра DICOM-снимков на сервер (листинг 24).

Листинг 24 – Метод отправки данных о задаче на сервер

```
1   public function setTaskInDbAction() {
2           if($this->getRequest()->isPost()){
3               $hash = $_POST['task_hash'];
4               $arr = $this->
5   ORM->getRepository
6   ('\Application\Entity\Tasks')->findBy([
7                   'task_hash' => $hash,
8               ]);
9               if(!count($arr)) {
10
11                  $pathToDir =
12                  '/var/www/dicomserver/public/' . $hash;
13                  mkdir($pathToDir, 0777);
14                  if
15  (move_uploaded_file($_FILES['file_1']['tmp_name'],
16  $pathToDir . '/normalize.txt'))
17  {
18                      $task = new Tasks();
19                      $task->setIsBusy(0);
20                      $task->setIsFinished(0);
21                      $task->setTaskHash($hash);
22                      $this->ORM->persist($task);
23                      $this->ORM->flush();
24                      return new JsonModel([
25                          'state' => 1,
26                          'message' => 'OK',
27                      ]);
28                  } else {
29                      return new JsonModel([
30                          'state' => 2,
31                          'message' => 'file do not
32                                        upload',
```

Окончание листинга 24

```
33                      ]);
34                  }
35              }
36              else {
37                  return new JsonModel([
38                      'state' => 3,
39                      'message' => 'this task already
40                                    exist',
41                  ]);
42              }
43          }
44          else
45              exit();
46  }
```

В сроке 2 встроенными средствами Zend Framework производится проверка является ли данный запрос к серверу POST-запросом. Если это не так, то в строке 42 отработает оператор «**else**» и метод завершит свою работу.

В строке 3 в переменную «**$hash**» помещается параметр POST-запроса «**task_hash**», который был сгенерирован в клиенте вызывающим кодом.

В строках с 4 по 8 производится выборка из БД по полю «**task_hash**» совпадающему с параметром текущего запроса, для проверки если ли уже в базе задача с таким же хэшем (строка 9). Если такая задача найдена, то в строках с 34 по 39 будет сформирован JSON-ответ, содержащий соответствующее сообщение об ошибке.

В строке 11 формируется строка, содержащая пусть до новой директории, которая предназначается для хранения файлов задачи и, в строке 13, с помощью метода «**mkdir**» создаётся данный каталог.

В строках с 14 по 16 производится загрузка файла, содержащего информацию об анализируемым DICOM-снимке, из POST-запроса. Функция загрузки файла вложена в условный оператор «if», так как метод сразу возвращает логическое значение об успехе операции. Если файл загрузить не удалось, в строках с 28 по 33 сформируется соответствующий JSON-ответ.

После загрузки файла, в строках с 18 по 23 будет сформирована новая запись в БД и вызывающий код получит положительный ответ от сервера (строки с 24 по 27).

Функция «setReadyAnalysisAction» предназначена для загрузки результирующих данных после анализа алгоритмом машинного обучения на сервер в каталог задачи (листинг 25).

Листинг 25 – Метод завершения работы над анализом

```
1   public function setReadyAnalysisAction(){
2       if($this->getRequest()->isPost()){
3           $hash = $_POST['source'];
4           if(isset($hash)) {
5               $taskReady =
6               $this->ORM->getRepository
7               ('\Application\Entity\Tasks')->findBy([
8                   'task_hash' => $hash,
9               ]);
10              if(count($taskReady)) {
11                  $pathToDir =
12  '/var/www/dicomserver/public/' . $hash;
13                  if
14  (move_uploaded_file($_FILES['file']['tmp_name'],
15  $pathToDir . '/pred.txt')) {
16
17                      $taskReady = array_pop($taskReady);
18                      $taskReady->setIsBusy(0);
19                      $taskReady->setIsFinished(1);
20                      $this->ORM->persist($taskReady);
21                      $this->ORM->flush();
22
23                      return new JsonModel([
24                          'state' => 1,
25                          'message' => 'OK',
26                      ]);
27
28                  } else {
29                      return new JsonModel([
30                          'state' => 2,
31                          'message' =>
32                          'file do not upload',
33                      ]);
34                  }
35
36              }
```

Окончание листинга 25

```
37
38              else {
39                  return new JsonModel([
40                      'state' => 3,
41                      'message' =>
42                  'task with this hash do not exist',
43                  ]);
44              }
45
46          }
47
48          else {
49              return new JsonModel([
50                      'state' => 4,
51                      'message' =>
52                      'source param did not set',
53                  ]);
54          }
55  }
56      else
57          exit();
58  }
```

В сроке 2 встроенными средствами Zend Framework производится проверка является ли данный запрос к серверу POST-запросом. Если это не так, то в строке 57 отработает оператор «**else**» и метод завершит свою работу.

В строке 3 в переменную «**$hash**» помещается параметр POST-запроса «**source**», который был заранее передан в python-приложение, для идентификации задачи. Если данная переменная не была передана в POST-запросе (строка 4), то в строках с 48 по 53 сформируется соответствубщий JSON-ответ.

В строках с 5 по 9 производится выборка из БД по полю «**task_hash**» совпадающему с параметром текущего запроса, для проверки существует ли данная задача в БД (строка 10). Если такая задача не найдена, то в строках с 39 по 43 будет сформирован JSON-ответ, содержащий соответствующее сообщение об ошибке.

В строках с 11 по 12 формируется переменная, содержащая пусть до каталога с файлами задачи, куда будет помещён файл с результатами анализа.

В строках с 14 по 15 производится загрузка файла, содержащего результат анализа DICOM-снимка, из POST-запроса. Функция загрузки файла вложена в условный оператор «if», так как метод сразу возвращает логическое значение об успехе операции. Если файл загрузить не удалось, в строках с 29 по 33 сформируется соответствующий JSON-ответ.

После загрузки файла, в строках с 17 по 21 будет изменена запись в БД (присвоен статус «**IsFinished**»), соответствующая данной проанализированной задаче и вызывающий код получит положительный ответ от сервера (строки с 23 по 26).

Функция «filePredictionReadyAction» предназначена для опроса сервера клиентом о наличии файла с результатами анализа изображения (листинг 26).

Листинг 26 – Запрос результата анализа

```
1   public function filePredictionReadyAction(){
2      if($this->getRequest()->isGet()) {
3      $hash = $_GET['hash'];
4      $taskReady =
5         $this->ORM->getRepository(
6         '\Application\Entity\Tasks')->findBy([
7                   'task_hash' => $hash,
8               ]);
9      if(count($taskReady)) {
10      if(file_exists(
11         '/var/www/sf_dicomserver/public/' .
12         $hash . '/pred.txt')){
13             return new JsonModel([
14                         'state' => 1,
15                         'message' => 'OK',
16                    ]);
17      }
18     else
19        return new JsonModel([
20                   'state' => 2,
21                   'message' =>
22                   'prediction file do not exist',
23                   ]);
24     }
25     else {
```

Окончание листинга 26

```
26          return new JsonModel([
27                      'state' => 3,
28                      'message' =>
29                      'task with this hash do not exist',
30                     ]);
31           }
32      }
33      else
34          exit();
35  }
```

В строке 2 встроенными средствами Zend Framework производится проверка, что данный обращение к методу является GET-запросом.

По параметру «**hash**» (строка 3) производится выборка записи из БД, содержащей информацию о запрашиваемой задаче (строки с 5 по 8). Если хэш-имя задачи передано без ошибок (строка 9), то будет проведена проверка, существует ли файл с результатами анализа от нейронной сети (строк и с 11 по 17). Если данного файла нет, то будет передан соответствующий ответ вызывающему коду (строки с 19 по 23).

Метод «consolestarttaskAction» предназначен для автоматического старта python-приложения для анализа DICOM-снимков (листинг 28). Данный метод запускается планировщиком заданий CRON раз в 5 минут (листинг 27).

Листинг 27 – Скрипт для планировщика CRON

```
1   */5 * * * * php /var/www/dicomserver/public/index.php
2   consolestarttask
```

Листинг 28 – Метод запуска анализа

```
1   public function consolestarttaskAction(){
2           $req = $this->getRequest();
3           if($req instanceof ConsoleRequest){
4               $arrWithTasks =
5               $this->ORM->getRepository
6               ('\Application\Entity\Tasks')->findBy([
7                   'is_busy' => 1,
8               ]);
9               if(!count($arrWithTasks)){
10                  $arrWithTasks =
11                  $this->ORM->getRepository
12                  ('\Application\Entity\Tasks')->findBy([
```

Окончание листинга 28

```
13                      'is_busy' => 0,
14                      'is_finished' => 0,
15                  ]);
16                  $arrWithTasks = $arrWithTasks[0];
17                  $arrWithTasks->setIsBusy(1);
18                  $this->ORM->persist($arrWithTasks);
19                  $this->ORM->flush();
20
21                  shell_exec(
22           'sh /var/www/dicomserver/call_docker.sh -p ' .
23           $arrWithTasks->getTaskHash());
24              }
25          }
26          else
27              exit();
28      }
```

В строках 2 и 3 производится проверка является ли данный запрос к методу пришедшем из терминала Ubuntu, то есть функция была вызвана планировщиком заданий CRON. Иначе, метод завершит свою работу (строка 27).

Далее необходимо проверить, что в данный момент ни одна задача не находится в состоянии анализа, для этого в строках с 4 по 8 производится выборка из БД по полю «**is_busy** = 1». Если подобные записи не будут найдены (строка 9), то, в строках с 10 по 15 из БД произведется выборка со всеми задачами, которые имеют статус «Не проанализирован» (is_finished = 0). Затем, самая первая запись из данной получит статус «is_busy = 1» (строки с 16 по 19).

Для запуска python-приложения для анализа используется sh-скрипт, который вызывается с помощью php-метода «**shell_exec**» с параметром, являющимся уникальным хэш-значением задачи.

Выводы по главе 3

В главе 3 подробно рассмотрен процесс установки и настройки среды web-сервера. Приведены описание модели данных, ее связь с БД. Описаны функции и алгоритмы web-приложения для передачи данных между программной-клиентом для просмотра DICOM-снимков и алгоритмом анализа, использующего методы машинного обучения.

ЗАКЛЮЧЕНИЕ

В выпускной квалификационной работе представлена реализация клиент-серверного программного обеспечения для просмотра и анализа DICOM-снимков; были подробно показаны и описаны реализация и функциональность ПО.

В работе решены следующие задачи:

- проведен обзор существующих программных продуктов для работы со снимками формата DICOM;
- проведен обзор и анализ существующего инструментария для реализации ПО для работы с DICOM-изображениями;
- реализована клиентская часть ПО, со следующей функциональностью:
  - выбор DICOM-директории, просмотр серий снимков;
  - выбор изображения, требующего анализа, предобработка и отправка на сервер.
- реализована серверная часть ПО, выполняющая следующие задачи:
  - агрегирование информации об изображениях, полученных от клиента;
  - запись в БД информации о задачах и предоставление записей клиенту;
  - взаимодействие с программной для анализа DICOM-изображений: запуск алгоритма и получение результата.

Новизна разработанного программного обеспечения состоит в том, что оно может быть использовано как система принятия решений в обучении молодых специалистов.

Полная схема взаимодействия всех компонент программного обеспечения представлена на рисунке 41.

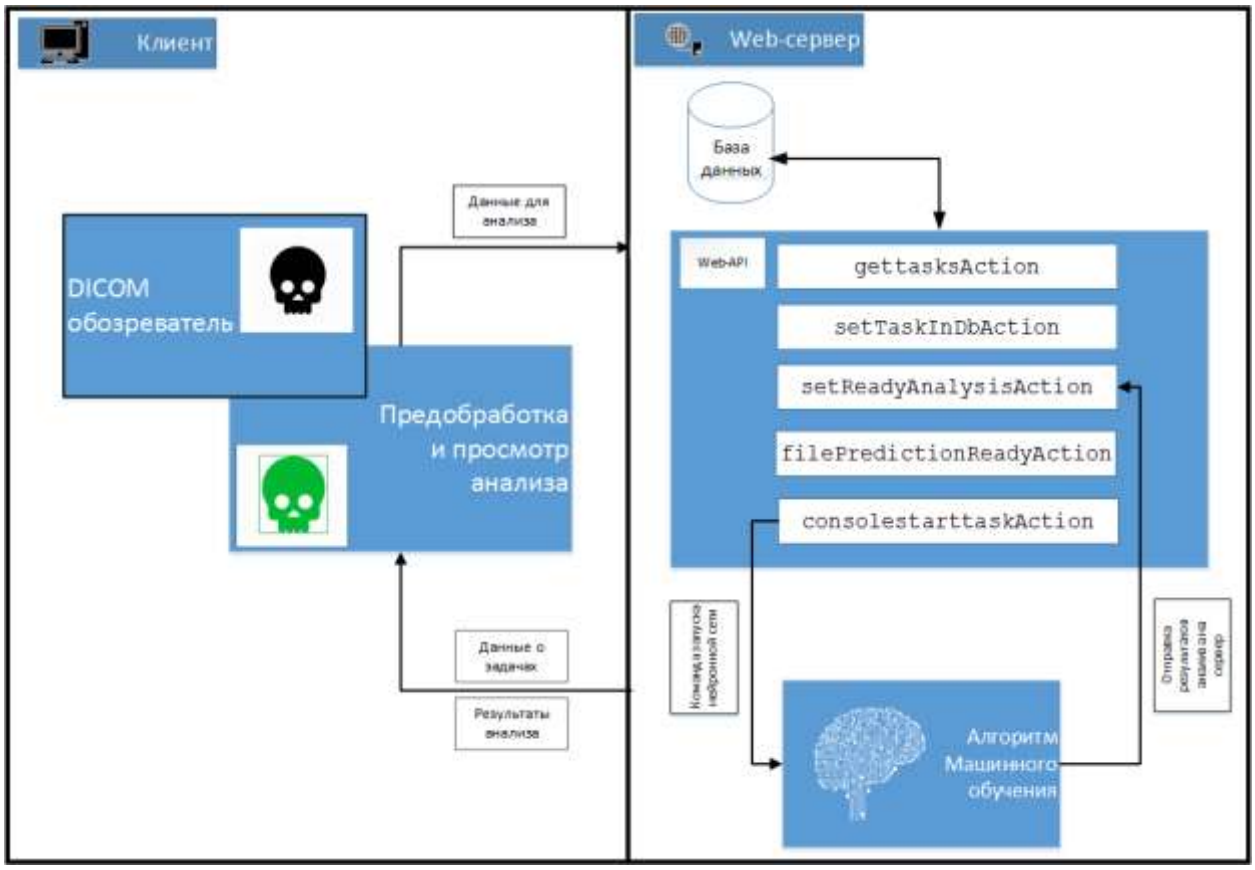

Рисунок 41 – Схема взаимодействия компонент ПО

В качестве направлений дальнейшего усовершенствования прикладного-программного обеспечения следует рассматривать следующие:

- реализация анализа DICOM-изображений для любого из органов;
- усовершенствование серверной части ПО для одновременного анализа более одной задачи;
- расширение функционала обмена DCIOM-снимками;
- создание более удобного пользовательского интерфейса клиента.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

S.Blair J. The Biomedical Engineering handbook. 1995. 2650-2659 pp.

.

Курс-АС1 Н.п.п. Стандарт DICOM 3.0 [Электронный ресурс] // Курс-АС1: . [сайт]. [1993]. URL: http://www.course-as.ru/dicomdoc.html (дата обращения: 26.11.2016).

ВСА. Стандарт DICOM [Электронный ресурс] // ВСА электроник: [сайт]. . URL: http://mri.com.ua/page/text/name=dicom (дата обращения: 26.11.2016).

Pöppl S.J. Diploma thesis "Using DICOM SR in Pathology" [Электронный . ресурс] // University of Lubeck: [сайт]. [2012]. URL: http://www.schoech.de/ diploma/toolkits.html (дата обращения: 04.12.2016).

Clunie D.A. DICOM Information Sources [Электронный ресурс] // David . Clunie's Medical Image Format Site: [сайт]. [2016]. URL: http:// www.dclunie.com/medical-image-faq/html/part8.html (дата обращения: 11.12.2016).

Crabb A. Display DICOM [Электронный ресурс] // I do imaging: free medical . software: [сайт]. [2002]. URL: http://www.idoimaging.com/index.shtml (дата обращения: 11.12.2016).

Medixant. RadiAnt DICOM Viewer [Электронный ресурс] // RadiAnt DICOM . Viewer: [сайт]. [2009]. URL: http://www.radiantviewer.com/ (дата обращения: 18.12.2016).

ИНОБИТЕК. Inobitec DICOM Viewer [Электронный ресурс] // Inobitec . DICOM Viewer: [сайт]. URL: http://www.inobitec.ru/ (дата обращения: 18.12.2016).

DICOM O. Online DICOM Viewer [Электронный ресурс] // Online DICOM . Viewer: [сайт]. [2016]. URL: https://www.onlinedicomviewer.com (дата обращения: 18.12.2016).

dcm4che. Weasis DICOM Viewer [Электронный ресурс] // Weasis DICOM
0. Viewer: [сайт]. [2001]. URL: https://dcm4che.atlassian.net/wiki/display/WEA/Home (дата обращения: 18.12.2016).

Pixmeo. OsiriX [Электронный ресурс] // OsiriX: [сайт]. [2003]. URL: http://
1. www.osirix-viewer.com/ (дата обращения: 18.12.2016).

Santesoft. Sante DICOM Viewer 3D [Электронный ресурс] // Sante DICOM
2. Viewer 3D: [сайт]. [1999]. URL: http://www.santesoft.com/win/sante_dicom_viewer_3d_free/sante_dicom_viewer_3d_free.html (дата обращения: 18.12.2016).

Gnandt A. openDICOM.NET [Электронный ресурс] // openDICOM.NET:
3. [сайт]. [2006]. URL: http://opendicom.sourceforge.net/index.html (дата обращения: 12.18.2016).

Hafey C. Cornerstone [Электронный ресурс] // Cornerstone: [сайт]. [2015].
4. URL: https://github.com/chafey/cornerstone (дата обращения: 18.12.2016).

fo-dicom_contributors. Fellow Oak DICOM [Электронный ресурс] // Fellow
5. Oak DICOM: [сайт]. [2012]. URL: https://github.com/fo-dicom/fo-dicom (дата обращения: 18.12.2016).

Medical_Connections. DICOM Objects [Электронный ресурс] // DICOM
6. Objects: [сайт]. [2001]. URL: http://www.medicalconnections.co.uk/DicomObjects (дата обращения: 18.12.2016).

Malaterre M. Grassroots DICOM [Электронный ресурс] // Grassroots DICOM :
7. [сайт]. [2003]. URL: http://gdcm.sourceforge.net/wiki/index.php/Main_Page (дата обращения: 18.12.2016).

LEADTOOLS. LEADTOOLS [Электронный ресурс] // LEADTOOLS: [сайт].
8. [1990]. URL: https://www.leadtools.com/ (дата обращения: 18.12.2016).

Gobbi D. vtk-dicom [Электронный ресурс] // vtk-dicom: [сайт]. [2012]. URL:
9. https://github.com/dgobbi/vtk-dicom (дата обращения: 18.12.2016).

Technology M. MST DICOM SDK [Электронный ресурс] // MS Technology:

0. [сайт]. [1991]. URL: http://www.ms-technology.com/toolkits/dicom/ (дата обращения: 18.12.2016).

1. Оснащенность компьютерным оборудованием Российских медицинских организаций [Электронный ресурс] // КОМПЛЕКСНЫЕ МЕДИЦИНСКИЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ: [сайт]. [2018]. URL: http://www.kmis.ru/blog/osnashchennost-kompiuternym-oborudovaniem-rossiiskikh-meditsinskikh-organizatsii (дата обращения: 15.05.2018).

2. Общие сведения о платформе.NET Framework [Электронный ресурс] // docs.microsoft.com: [сайт]. [2018]. URL: https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/overview (дата обращения: 15.05.2018).

3. Общие сведения о Windows Forms [Электронный ресурс] // docs.microsoft.com: [сайт]. [2018]. URL: https://docs.microsoft.com/ru-ru/dotnet/framework/winforms/windows-forms-overview (дата обращения: 15.05.2018).

4. Обзор интегрированной среды разработки Visual Studio [Электронный ресурс] // docs.microsoft.com: [сайт]. [2018]. URL: https://docs.microsoft.com/ru-ru/visualstudio/ide/visual-studio-ide (дата обращения: 15.05.2018).

5. O. R. F.P.B.T. U-Net: Convolutional Networks for Biomedical Image Segmentation // Computer Science Department and BIOSS Centre for Biological Signalling Studies, 2015.

6. Medixant. RadiAnt DICOM Viewer [Электронный ресурс] // RadiAnt DICOM Viewer: [сайт]. [2009]. URL: http://www.radiantviewer.com/ (дата обращения: 15.05.2018).

7. Converting CT Data to Hounsfield Units [Электронный ресурс] // idlcoyote: [сайт]. [2018]. URL: http://www.idlcoyote.com/fileio_tips/hounsfield.html (дата обращения: 15.05.2018).

8. DICOM Rescale Intercept / Rescale Slope and ITK [Электронный ресурс] // blog.kitware: [сайт]. [2018]. URL: https://blog.kitware.com/dicom-rescale-

intercept-rescale-slope-and-itk/ (дата обращения: 15.05.2018).

Фреймворк [Электронный ресурс] // Wikipedia: [сайт]. [2018]. URL: https://
9. ru.wikipedia.org/wiki/Фреймворк (дата обращения: 15.05.2018).

About [Электронный ресурс] // Zend Framework: [сайт]. [2018]. URL: https://
0. framework.zend.com/about (дата обращения: 15.05.2018).

Getting Started [Электронный ресурс] // Dependency Manager for PHP: [сайт].
1. [2018]. URL: https://getcomposer.org/doc/00-intro.md (дата обращения: 15.05.2018).

Model-View-Controller [Электронный ресурс] // Wikipedia: [сайт]. [2018].
2. URL: https://ru.wikipedia.org/wiki/Model-View-Controller (дата обращения: 15.05.2018).

Object Relational Mapper [Электронный ресурс] // Doctrine: [сайт]. [2018].
3. URL: https://www.doctrine-project.org/projects/orm.html (дата обращения: 15.05.2018).

JSON [Электронный ресурс] // Wikipedia: [сайт]. [2018]. URL: https://
4. ru.wikipedia.org/wiki/JSON (дата обращения: 15.05.2018).

ПРИЛОЖЕНИЕ А

Выявление соответствия требованиям готовых программных продуктов для работы с DICOM

Таблица А.1

RadiAnt DICOM Viewer	Inobitec DICOM Viewer	
+	+	
+	+	
+	+	
+	+	
-	-	
-	-	

ПРИЛОЖЕНИЕ Б

Выявление соответствия требованиям программных библиотек для работы с DICOM

Таблица Б.1

Требуемые признаки	openDICOM (C#)	Cornerstone (JS)	FO-DICOM (C#)	dcm4che (JAVA)	Dicom Objects (C#)	GDCM (Python, C#, Java, C++)
Загрузка 3D-снимка формата *.dicom	-	-	-	+/-	+	-
Навигация по снимку, масштабирование	+	+	-	+	+	-
Детализация путём выбора среза	-	-	-	-	+	-
Инструментарий для работы со снимком: поиск расстояния между точками, поиск площади и периметра выделенной области, измерение угла	-	-	-	+	+	-
Наличие API для внедрения в другие программные продукты	+	+	+	+	+	+
Бесплатность приложения	+	+	+	+	-	+

ПРИЛОЖЕНИЕ В

Создание задачи анализа выбранного изображения

```
1   private void buttonSaveCurrentPlane_Click(object sender,
2   EventArgs e)
3      {
4         Random RandNum = new Random();
5         string HashForDir = checkMD5(
6         DateTime.Now.ToLongDateString() +
7         Convert.ToString(RandNum.Next(1111, 9999)));
8         Directory.CreateDirectory(Program.TempDir +
9         "\\" + HashForDir);
10        StreamWriter IniFileInDir =
11        File.CreateText(Program.TempDir + "\\" +
12        HashForDir + "\\Ini.ini");
13        IniFileInDir.WriteLine("[Info]");
14        IniFileInDir.WriteLine(@"temp_folder="+
15        Program.TempDir +"\n" +
16        "max_scalar = \"\"\n"+
17        "min_scalar = \"\"\n"+
18        @"dicom_folder =" + Path.GetDirectoryName(
19        openDicomFileDialog.FileName ));
20        IniFileInDir.Close();
21        Program.IniObj.IniWriteValue("Info",
22        "curr_work_dir", HashForDir);
23        Ini localIni = new Ini(Program.TempDir + "\\"
24        + HashForDir + "\\Ini.ini");
25        StreamWriter streamWriter =
26        new StreamWriter(Program.TempDir + "\\" +
27        HashForDir + "\\" + "imageScalars.txt");
28        for (int i = 0; i < 500; i++)
29        {
30           for (int j = 0; j < 500; j++)
31           {
32              streamWriter.Write(Convert.ToString(
33              SiteplanesObj.planeWidgetZ.
34              GetResliceOutput().
35              GetScalarComponentAsDouble(i, j, 0, 0)
36              ) + " ");
37           }
38                 streamWriter.Write("\n");
39        }
40        streamWriter.Close();
41        vtkWindowToImageFilter imf =
42        new vtkWindowToImageFilter();
43        imf.SetInputBufferTypeToRGBA();
44        switch (currentSliceDimension)
45        {
46           case 1:
47           {
48               imf.SetInput(
```

```
49                  vtkFormsWindowControlX.GetRenderWindow());
50                  localIni.IniWriteValue("Info",
51                  "max_scalar",Convert.ToString(
52                  SiteplanesObj.rangeXMax ));
53                  localIni.IniWriteValue("Info",
54                  "min_scalar", Convert.ToString(
55                  SiteplanesObj.rangeXMin));
56                  break;
57              }
58              case 2:
59              {
60                 imf.SetInput(
61                 vtkFormsWindowControlY.GetRenderWindow());
62                 localIni.IniWriteValue("Info",
63                 "max_scalar", Convert.ToString(
64                 SiteplanesObj.rangeYMax));
65                 localIni.IniWriteValue("Info",
66                 "min_scalar", Convert.ToString(
67                 SiteplanesObj.rangeYMin));
68                 break;
69              }
70              case 3:
71              {
72                 imf.SetInput(
73                 vtkFormsWindowControlZ.GetRenderWindow());
74                 localIni.IniWriteValue("Info",
75                 "max_scalar", Convert.ToString(
76                 SiteplanesObj.rangeZMax));
77                 localIni.IniWriteValue("Info",
78                 "min_scalar", Convert.ToString(
79                 SiteplanesObj.rangeZMin));
80                 break;
81              }
82          }
83          vtkPNGWriter pngw =
84          new vtkPNGWriter();
85          pngw.SetFileName(Program.TempDir +
86          "\\" + HashForDir + "\\" +
87          "\\image.png");
88          pngw.SetInputConnection(imf.GetOutputPort());
89          pngw.Write();
90          System.Diagnostics.Process.Start(
91          AppDomain.CurrentDomain.BaseDirectory +
92          "imagePreprocessing.exe");
93      }
```

ПРИЛОЖЕНИЕ Г

Паддинг изображения

```
1   public Bitmap paddingReflect(Bitmap image,
2   int reflectParam)
3   {
4      Bitmap resBtm = new Bitmap(
5      image.Width + reflectParam,
6      image.Height + reflectParam);
7      for (int i = 0; i < image.Width; i++)
8      {
9         for (int j = 0; j < image.Height; j++)
10        {
11           resBtm.SetPixel(i + reflectParam / 2,
12           j + reflectParam / 2, Color.FromArgb(
13           Convert.ToInt32(image.GetPixel(i, j).R),
14           Convert.ToInt32(image.GetPixel(i, j).R),
15           Convert.ToInt32(image.GetPixel(i, j).R)));
16        }
17     }
18     //top
19     for (int i = 0; i < image.Width; i++)
20     {
21        for (int j = 0; j < reflectParam / 2; j++)
22        {
23           resBtm.SetPixel(i + reflectParam / 2,
24           reflectParam / 2 - j,
25           Color.FromArgb(
26           Convert.ToInt32(image.GetPixel(i, j).R),
27           Convert.ToInt32(image.GetPixel(i, j).R),
28           Convert.ToInt32(image.GetPixel(i, j).R)
29           ));
30        }
31     }
32     //left
33     for (int i = 0; i < reflectParam / 2; i++)
34     {
35        for (int j = 0; j < image.Height; j++)
36        {
37           resBtm.SetPixel(reflectParam / 2 - i,
38           j + reflectParam / 2,
39           Color.FromArgb(
40           Convert.ToInt32(image.GetPixel(i, j).R),
41           Convert.ToInt32(image.GetPixel(i, j).R),
42           Convert.ToInt32(image.GetPixel(i, j).R)
43           ));
44        }
45     }
46     //right
47     for (int i = image.Width - reflectParam / 2;
48          i < image.Width; i++)
```

```
49      {
50          for (int j = 0; j < image.Height; j++)
51          {
52              resBtm.SetPixel(resBtm.Width -
53              (i - (image.Width - reflectParam / 2)) - 1,
54              j + reflectParam / 2,
55              Color.FromArgb(
56              Convert.ToInt32(image.GetPixel(i, j).R),
57              Convert.ToInt32(image.GetPixel(i, j).R),
58              Convert.ToInt32(image.GetPixel(i, j).R)
59              ));
60          }
61      }
62      //bottom
63      for (int i = 0; i < image.Width; i++)
64      {
65          for (int j = image.Height - reflectParam / 2;
66               j < image.Height; j++)
67          {
68              resBtm.SetPixel(i +
69              reflectParam / 2, resBtm.Height -
70              (j - (image.Height - reflectParam / 2)) - 1,
71              Color.FromArgb(
72              Convert.ToInt32(image.GetPixel(i, j).R),
73              Convert.ToInt32(image.GetPixel(i, j).R),
74              Convert.ToInt32(image.GetPixel(i, j).R)
75              ));
76          }
77      }
78      return resBtm;
79  }
```